Forj Documentation

Release 1.0

Forj community

May 01, 2015
Contents

1 Documentation 3
  1.1 User guide ................................. 3
  1.2 Developer guide ............................ 22

2 Indices and tables ........................................ 37
Forj provides software developers with a rapid, flexible and open continuous integration (CI)/continuous delivery (CD) development platform. This environment is often referred to as a software forge, even though, at the time the name was invented, the notion of CI and CD was not as prevalent.

Forj provides:

- **A collection of pre-integrated continuous integration and delivery tool stacks called blueprints**
  - Forj has currently one continuous integration stack which mimics the OpenStack™ Open Source project. We look forward to build a Forj community to add many continuous integration stacks to the catalog.

- An orchestration and administration component called *Maestro* which provides a uniform way to provision projects and users as well as manage the forge (backup/restore, monitoring, updates).

See the [Forj web site](#) for more information, use cases and flows.
The documentation has 2 parts:

- The “user guide” section is relevant for Forj users as well as forge administrators and users.
- The “developer guide” is for anyone who wants to contribute to the Forj project.

### 1.1 User guide

#### 1.1.1 Introduction

**Organization**

Forj is combination of 2 components:

- **Maestro.** Maestro’s code is managed in *Forj’s forge*, and published to Github under *forj/maestro*.
- **Blueprint(s).** Read by Maestro to install a *forge*. Those blueprints are exposed in Forj’s catalog, developed in *Forj’s forge*, and published to Github under *forj*. There is one GIT repository per blueprint.

**Philosophy**

Forj is **vendor-agnostic**. The goal is to provide integrated software development tools which solve a problem. The tools can be all Open Source, a mix of Open Source and proprietary tools, or a set of proprietary tools. As long as a tool has a way to be installed or integrated with other tools, Forj will have a way to make it part of the *Forge*. In other words, it is possible to integrate tools hosted in the cloud with tools installed on your own servers to create a *forge*.

Forj is **destination-agnostic**. The *forge* can be installed on a public or a private cloud or on premises. Forj leverages the *fog library* to be able to provision systems on many different clouds and thus avoiding vendor lock-in. A great benefit is that you can *relocate* a forge from a public cloud to a private cloud or from a cloud provider to another.

#### 1.1.2 Installation

There are 2 ways to provision a forge: through forj command line or the forj portal.
Forj Documentation, Release 1.0

Forj command line

You use the forj command line to provision forges on a private cloud or a public cloud of your choice. The command line also helps you integrate forge creation and management in your own environment. For example, if you have a catalog of services that you offer, and that you want to make forges available, the command line will help you achieve this objective. Finally, the command line is also how you can create your own blueprint or edit an existing one. See the developer guide for more information on developing for Forj.

For more information include installation and man page on the Forj CLI page.

Forj portal

You can use Forj’s portal to install a forge on HP Public Cloud or an OpenStack compatible public cloud. If you are an enterprise, you can also make forj available through your IT services portal. As an enterprise, you can replicate such portal and offer “forges as a service”. Forj’s portal uses HP CSA product.

The portal will ask you the credentials of your cloud so that it can provision the forge automatically. If you are not comfortable with this, or want to install forges in a private cloud, you should use “Forj command line”.

1.1.3 Administration

You administer a forge through the Maestro user interface.
This includes:

- General forge administration (health, backup & restore)
- Project administration
- Tool administration
- User administration

Tools

The list of tools that you see in the Maestro User Interface depends on the blueprint. Each box, which represents a tool exposed can be clicked to open the tool itself.

- Admin: if you are an administrator of the forge, tool which have one will expose an additional “Admin” link, to jump to administration interface of that tool.
- Workload: provides detail on the tool workload, such as CPU, Memory or storage consumption of the tool, as well as tool specific metrics (eg commits / day)
- Backup: tool specific backup and restore operations
Projects

The notion of projects varies from blueprint to blueprint. In the “Redstone” blueprint, the project creation process from OpenStack infra is automated and exposed with the Maestro UI. Please refer to the appropriate section of project management for the blueprint you use.

By default this section is only available for the forge administrator, but you can change that if you like currently we have 3 levels of access:

- Administrator (They have access to everything in the forge)
- Authenticated Users (Limited access to a few features of the forge)
- Anonymous Users (Only view access to the tools of the forge)

To change the access level for the Projects section you need to modify the file maestro.yaml that is located in:

`[forj-config]/modules/runtime_project/hiera/layouts/maestro.yaml`

Update or add the yaml configuration section ‘jimador::site’, example:

```
jimador::site:
  ...  
  global_manage_projects: "admin"
```

Available value options for global_manage_projects include:

- “admin” = administrator access only
- “authenticated” = authenticated users and administrator
- “anonymous” = everyone

Commit your changes for `[forj-config]`, approve them, then apply your puppet manifest on maestro, or allow at least two runs for puppet to update. The file `/opt/config/production/config.json` will be updated with your settings:

```
FACTERLIB="/var/lib/puppet/lib/facter"
PATH="/usr/sbin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/bin"
puppet agent --test
puppet agent --test  # run this at least twice for updates to get propagated
```

If you want to see if your change was applied open the config.json file and there you will see the “global_manage_projects” with your new value.

Users

Like projects, users may be managed differently from blueprint to blueprint.

Note: Implementation is in progress.

Backup/Restore

Maestro provides backup and restore capabilities for the forge. There is the notion of a forge wide backup/restore process, as well as per tool backup/restore. The forge wide backup status and restore operation is available to administrators through Maestro UI.

Note: Implementation is in progress. While backups are implemented, the status, and the ability to restore is not exposed in Maestro UI.
Monitoring

Each blueprint can consume monitoring services and expose jauges in the Maestro UI.

Note: Implementation is in progress.

1.1.4 Redstone

Summary

If you want to develop like the OpenStack ® project, the “Redstone” blueprint is what you are after.

Learn more on the processes that are used by the OpenStack project here:

- How to contribute to OpenStack
- OpenStack infrastructure team

Check this tutorial which shows Redstone in action:

---

[1] The OpenStack Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation’s permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.
Tools and features

Very high level, the Redstone flow looks like this:

In its current version, the Redstone blueprint contains the following features:

- Puppet master automation, create new modules and promote them in the forj-config project.

- Gerrit/git services for revision control and change management.
  - Login with Launchpad openid, use www.launchpad.net to setup your email account for authentication.
  - Create open commits that can be reviewed with the team and gated for change before they hit master.
  - Tag and release changes when they are ready for testing or production.
  - Mark commits as work in progress or abandon revert old changes.
  - Create new gerrit projects and acl?s managed from source in forj-config project.

- Jenkins/Zuul integration
  - Create zuul macros to automate common build task, or re-use pre-existing openstack macros included in the pre-configured forj-config project.
  - Create automatic Jenkins jobs so that your team no longer manages Jenkins setup on Jenkins master, all configuration is managed as source in forj-config git project.
  - Gate changes and execute automated test as soon as changes make it to gerrit/git.
  - Promote changes based on gerrit events for tagging or comments made during reviews.
  - Login with launchpad openid
  - Zuul status: see what changes are being automatically tested, the progress and the status from one interface

- Pastebin integration
  - Collaborate with team mates with code snippets and log pasting for troubleshooting defects.

- Defect tracking
  - Mark gerrit changes with defect id?s to link to your defect system with Launchpad or HP Agile Manager

Project Management

You can manage project using Maestro user interface, or by editing the “forj-config” GIT repository (through Gerrit).

Adding a new project

In the Redstone blueprint, projects are managed by code, exactly like the OpenStack infrastructure project. This code, which sits in the forj-config Gerrit repository describes a project and is used to provision the entire chain associated to a project (Gerrit, Jenkins, Zuul).

With Maestro User Interface

- Enter the maestro UI and sign-in if needed, so you become the administrator and can add projects. The first user who logs becomes administrator and can add other administrators. Only administrator can create projects.
- Connect to Gerrit. The first user who logs in to Gerrit becomes administrator
As a Gerrit administrator, add yourself to the “forj-config” group. This allows you to review and approve changes in the forj-config project.

- Go to the projects tab then add projects icon, finally set the name for you project and wait ~15 minutes until the change is propagated to gerrit (this uses Puppet mechanisms).
- Go to Gerrit and approve (+2) or reject (-2) the project creation change.
- Once Puppet has done its work (up to 30 minutes), the new project is fully created and configured in Gerrit, Jenkins and Zuul.

**Note:** You can force “Puppet Apply” and speed up the process through the Jenkins “puppet-apply-all-nodes” job, or directly in a terminal session on the Maestro / Puppet master system.

**Using “Redstone” mechanisms** You have full control of project creation by checking out the forj-config project and modifying the files that you need.

**Note:** In the “Redstone” blueprint, everything is treated as code. You must go through Gerrit mechanisms to edit project configuration. Otherwise, your changes will be dismissed the next time the puppet master runs.

- **Check out the forj-config project** Each forge has its own forj-config project. Make sure your ssh keys are added to the gerrit server. To approve a change, make sure your user account is a member of the forj-core group.
- **Assigning someone to be a verifier for forj-config project** When there is no jenkins/zuul functionality on the forj-config project, it’s sometimes necessary to verify changes manually. This can be performed by updating the group membership for Continuous Integration Tools group in gerrit.
- **Add an acl file** (to the acls/production directory below) by using cookiecutter.config as an example found in the below path (i.e. make a copy):
  ```
  [forj-config.git] / modules / runtime_project / files / gerrit / acls / templates / cookiecutter.config
  ```
- **Place cookiecutter.config in directory** (projectname should match the name of the project): (note: change the ‘cookiecutter-core’ to whatever group name you would like to use in your project: example: my-new-proj1-core):
  ```
  [forj-config.git] / modules / runtime_project / files / gerrit / acls / production / projectname.config
  ```
- **To create project edit file**:
  ```
  [forj-config.git] / modules / runtime_project / templates / gerrit / config / production / review.projects.yaml.erb
  ```
- **Push changes to your gerrit repo**:
  ```
  $ git add <new-project-acl-file>
  $ git add review.projects.yaml.erb
  $ git commit -m "my new project"
  $ git push
  ```
- **To learn more on how to configure yaml**, see jeepb docs.
- **You can migrate public projects with the upstream option.**
- **Projects that are created in gerrit currently have no approach for deletion, but these can be removed from normal users view through acl changes. For more info, please refer to : rename project or remove project**
Adding a new jenkins job and configure Zuul for a given project in gerrit

Zuul configuration consist of 4 basic parts.

1. update hieradata to include any new templates that will be used for the job in run-time_project/files/hiera/hieradata/Debian/nodetype/ci.yaml
   • this is only needed if you need a new compiler option, or new tool that will not exist on the build server.
   • configure in the following section ci-node -> class cdk_project::jenkins -> job_builder_configs.
   Example:

   ```
   cdk_project::jenkins::job_builder_configs:
   - 'tutorials.yaml'
   - '<new_job_template_name>.yaml'
   ```

2. configure the new template into runtime_project/templates/jenkins_job_builder/config/
   • a pre-existing template file can be used to describe the builders for the job, or a new one can be created
   • pre-existing macros can be found in runtime_project/files/jenkins_job_builder/config/macros.yaml

3. update layout.yaml in runtime_project/files/zuul/config/production/layout.yaml
   • the projects section should be updated with the new project and gates, along with jobs that will be executed from projects.yaml, example:

   ```
   projects:
   - name: tutorials
     check:
     - tutorials-flake8
     gate:
     - tutorials-flake8
     post:
     - puppet-apply-all-nodes
     release:
     - tutorials-flake8
   ```

4. add the project section to runtime_project/files/jenkins_job_builder/config/projects.yaml
   • this will define the jobs to be created in jenkins, job names will be mapped to builders by zuul.
   The “name” must match the job-template layout file (line 2 in the jenkins_job_builder file), and the “git_project” must match with the name of your project in gerrit.

   ```
   projects:
   name: tutorials
   git_project: tutorials
   branch: master
   jobs:
   - '{name}-flake8'
   - '{name}-<new_job_name>'
   ```

Once this is done, you will need to push the changes to gerrit, verify and submit. Next the eroplus box will need to run puppet cycle, or puppet agent -t to get the new runtime_project updates. Finally the ci server will need to run a puppet cycle or puppet agent -t so that the job builder can setup the job.

Note: More info on zuul: http://ci.openstack.org/zuul
Remove a project in gerrit

- Stop gerrit:
  
  ```
  $ sudo service gerrit stop
  ```

- start the gsql client on local admin bash shell:
  
  ```
  $ java -jar /home/gerrit2/review_site/bin/gerrit.war gsql -d /home/gerrit2/review_site
  ```

- remove entries from table `account_project_watches`
  
  ```
  select * from account_project_watches;
  delete from account_project_watches where project_name = 'tutorials-2';
  ```

- Remove the repo from disk.
  
  ```
  $ rm -rf /var/lib/git/tutorials-2.git
  $ rm -rf /home/gerrit2/review_site/git/tutorials-2.git/
  ```

**Note:** this should be done on all replicas

- Start gerrit back up
  
  ```
  $ service gerrit start
  ```

User management

In the Redstone blueprint, the first user who authenticates to Gerrit and Jenkins become administrator. Then, it is the role of the administrator to add users in the respective tools and projects.

Configure Email Notifications

In order to gerrit send email notifications you need to configure it first.

- First you need to have at least one project configured in gerrit, and have a forj-config clone git repository.
- Have a external MTA account, in this example we will use sendgrid. This service has a free account option if you want to try it.
- In the path `forj-config/modules/runtime_project/files/hiera/layouts/` you need to make the following changes in the file `maestro.yaml`:

```
classes:
  - ....
  - exim_config
    exim_config::smarthost: 'smtp.sendgrid.net'
    exim_config::port: '587 byname'
    exim_config::smtp_require_auth: true
    exim_config::smtp_username: 'YOUR_USER'
    exim_config::smtp_password: 'YOUR_PASSWORD'
    exim_config::queue_smtp_domains: 'localhost.org'
    exim_config::relay_from_hosts:
```

1.1. User guide
In the same path as above, modify the file `review.yaml` adding the following lines:

```yaml
classes:
  - exim_config::utils
  - ...

  gerrit_config::smtpserver: "%{::exim_config::utils::maestro_ip}"
  gerrit_config::sendemail_from: 'YOUR_SENDFROM_EMAIL_NAME'
```

### Email Troubleshooting

If you are having problems with this configuration, you can try these steps to find where the problem is.

1. **Check if gerrit was configured correctly in Review server:**
   
   (a) Check gerrit’s config file located in `/home/gerrit2/review_site/etc/gerrit.config` you should have something similar:

   ```
   [sendemail]
   smtpServer = 10.0.0.90
   from = robot <robot@my.com>
   ```

   (a) A common problem is an empty IP address, just make sure that `[sendemail]` 
   
   (b) Check if the port 25 is open on maestro server:

   ```
   $ nc -v 10.0.0.90 smtp
   Connection to 10.0.0.90 25 port [tcp/smtp] succeeded!
   220 maestro.v5.dev.forj.io ESMTP Exim 4.76 Fri, 18 Jul 2014 22:00:05 +0000
   ```

   (a) Make sure that you have at least 2 users registered in gerrit before attempting to test it.

   (b) While you generate the email event in gerrit, keep monitoring log file: `tail -f /home/gerrit2/review_site/logs/error_log`, you should see if the mail was attempted to sent.

2. **Check if exim was correctly configured in maestro.**

   (a) Try to send a test email to check if the external smtp server was configured correctly:

   ```
   $ echo "Test email " | mail -s "test external" <email_address@my.com>
   ```

   (a) If you didn’t receive the mail you can test changing the parameters manually in exim config file

   ```
   $ service exim4 stop
   $ vim /etc/exim4/exim4.conf
   $ service exim4 start
   ```

   (a) As a alternative you can use a terminal email client to make further tests.

   ```
   $ apt-get install mutt
   $ vim ~/.muttrc
   ```
and add the following line:

```
set smtp_url = "smtp://smtp.sendgrid.net:587"
```

try to send a test email

```
echo "Test email " | mutt -s "test external" email_address@my.com
```

(a) You can view exim logs in the following location: `/var/log/exim4/mainlog`

- Allow puppet to run at least twice on both maestro and review nodes in order to see your changes show up.

## Configure DNS Names

Initially when you deploy your forge, you will reference your forge by ip address but this is not practical for your end-users. It’s possible to assign a well known dns name to your forge nodes from a DNS server outside of your cloud account. This section will only describe the areas of [forj-config] that can be updated to control the naming behavior for jenkins and gerrit. We will not attempt to describe how to use external DNS service.

1. The following files will be modified to give a well known domain name for each node. You should register the domains with your DNS provider.

<table>
<thead>
<tr>
<th>Directory</th>
<th>file</th>
<th>Proposed DNS names</th>
</tr>
</thead>
<tbody>
<tr>
<td>/files/hiera/layouts</td>
<td>maestro.yaml</td>
<td>maestro.yourdomain.com</td>
</tr>
<tr>
<td>/files/hiera/layouts</td>
<td>ci.yaml</td>
<td>ci.yourdomain.com</td>
</tr>
<tr>
<td>/files/hiera/layouts</td>
<td>review.yaml</td>
<td>review.yourdomain.com</td>
</tr>
<tr>
<td>/files/hiera/layouts</td>
<td>util.yaml</td>
<td>util.yourdomain.com</td>
</tr>
</tbody>
</table>

2. Configuration for **maestro** node.

<table>
<thead>
<tr>
<th>[forj-config]/modules/runtime_project/files/hiera/layouts/maestro.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
</tr>
<tr>
<td>jimador::site:</td>
</tr>
<tr>
<td>maestro_url: &quot;http://%{::maestro::node_vhost_lookup::vname}&quot;</td>
</tr>
<tr>
<td>#...</td>
</tr>
<tr>
<td>To:</td>
</tr>
<tr>
<td>jimador::site:</td>
</tr>
<tr>
<td>maestro_url: &quot;<a href="http://maestro.yourdomain.com">http://maestro.yourdomain.com</a>&quot;</td>
</tr>
<tr>
<td>#...</td>
</tr>
</tbody>
</table>

3. Configuration for **ci** node.

<table>
<thead>
<tr>
<th>[forj-config]/modules/runtime_project/files/hiera/layouts/ci.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
</tr>
<tr>
<td>#Jenkins tool</td>
</tr>
<tr>
<td>cdk_project::jenkins::vhost_name: &quot;%{::maestro::node_vhost_lookup::vname}&quot;</td>
</tr>
<tr>
<td>#...</td>
</tr>
<tr>
<td>#Zuul tool</td>
</tr>
<tr>
<td>cdk_project::zuul::vhost_name: &quot;%{::maestro::node_vhost_lookup::vname}&quot;</td>
</tr>
<tr>
<td>#...</td>
</tr>
</tbody>
</table>
cdk_project::zuul::zuul_url: "http://%{::maestro::node_vhost_lookup::vname}/p"
#

- To:

  #Jenkins tool
cdk_project::jenkins::vhost_name: "ci.yourdomain.com"
  #
  #Zuul tool
cdk_project::zuul::vhost_name: "ci.yourdomain.com"
  #
  cdk_project::zuul::zuul_url: "http://ci.yourdomain.com/p"
  #

4. Configuration for review node.

[forj-config]/modules/runtime_project/files/hiera/layouts/review.yaml

- From:

  #
  cdk_project::gerrit::vhost_name: "%{::maestro::node_vhost_lookup::vname}"
  cdk_project::gerrit::ip_vhost_name: "%{::maestro::node_vhost_lookup::vname}"
  cdk_project::gerrit::canonicalweburl: "https://%{::maestro::node_vhost_lookup::vname}/"
  #

- To:

  #
  cdk_project::gerrit::vhost_name: "review.yourdomain.com"
  cdk_project::gerrit::ip_vhost_name: "review.yourdomain.com"
  cdk_project::gerrit::canonicalweburl: "https://review.yourdomain.com/"
  #

5. Configuration for util node.

[forj-config]/modules/runtime_project/files/hiera/layouts/util.yaml

- From:

  #
  cdk_project::status::vhost_name: "%{::maestro::node_vhost_lookup::vname}"  
  cdk_project::status::graphite_url: "http://%{::maestro::node_vhost_lookup::vname}:8081"  
  cdk_project::status::static_url: "http://%{::maestro::node_vhost_lookup::vname}:8080"  
  cdk_project::status::maestro_url: "http://%{eroip}"

  # 'graphite' tool
cdk_project::graphite::vhost_name: "%{::maestro::node_vhost_lookup::vname}"  
  #
  # 'paste' tool
cdk_project::paste::vhost_name: "%{::maestro::node_vhost_lookup::vname}"  
  #

- To:

  #
  cdk_project::status::vhost_name: "util.yourdomain.com"
  cdk_project::status::graphite_url: "http://util.yourdomain.com:8081"  
  cdk_project::status::static_url: "http://util.yourdomain.com:8080"  
  cdk_project::status::maestro_url: "http://maestro.yourdomain.com"

  # 'graphite' tool
Install SSL Certificates

The Redstone blueprint will install a custom certificate that is digitally signed by a self signed certificate registry located on maestro. This is managed by the cacerts puppet module from the forj-oss/maestro repository. This however means that developers and end-users that access the review and ci nodes for your forge will be warned and prompted to continue navigation to the site because these certificates are not typically trusted by the browser.

In this section we will describe the process you can follow for configuring a digitally signed certificate from a certificate authority that would more commonly be trusted by your browser. This would replace the automated self signed certificates that are auto generated by maestro. When making changes to the [forj-config] repository, be aware that the same ci workflow used for updating projects and gates still applies here.

1. Generate a private key and certificate signing request(csr). This can be performed with these instructions documented on the verisign website.

   Example:

   ```
   $ openssl genrsa -passout pass:secretpass -des3 -out review.yourdomain.com.key 2048
   $ openssl req -new -key review.yourdomain.com.key -out review.yourdomain.com.csr
   ```

   • Submit the .csr request to your certificate authority (verisign for example) and save away your private keys. If you also recieve intermediate certificates, such as a corporate signing authority, you will need to save those as well. These will be your intermediate certificate chain.

   • Ideally this request is performed for review and ci nodes. Long term, you will also want one for the maestro node. At this time, we do not need a certificate for the maestro node.

2. Create a custom_certs module to deliver your certs.

   clone the [forj-config] project from your Redstone forge so that we can create a new module to store and deploy the certificates. Even though the private key is protected, it’s possible to store the credentials for the certificate in a secure hiera yaml file on maestro for decryption. In these steps we are only creating the certificates for review node, but you can repeat this for ci and maestro nodes.
Folder layout for custom_certs puppet module:

```
[forj-config]/modules/custom_certs/
```

<table>
<thead>
<tr>
<th>Directory</th>
<th>file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>./files/certs</td>
<td>chain.crt</td>
<td>This is your certificate signer public certs.</td>
</tr>
<tr>
<td>./files/certs</td>
<td>re-view.yourdomain.com.locked_key</td>
<td>This is the private key generated in step 1. Password included.</td>
</tr>
<tr>
<td>./files/certs</td>
<td>review.yourdomain.com.csr</td>
<td>Your certificate request</td>
</tr>
<tr>
<td>./files/certs</td>
<td>review.yourdomain.com.cst</td>
<td>Your x509 signed cert from your certificate provider.</td>
</tr>
<tr>
<td>./files/manifests</td>
<td>review_sslparams.pp</td>
<td>This will contain puppet ref vars to ssl cert contents.</td>
</tr>
</tbody>
</table>

3. Create a class file `<node>_sslparams.pp` for each cert that will be needed. In this case we create the class file for the `review` node (`review_sslparams.pp`). We can’t use the normal puppet file function because it will fail on runs where the cert doesn’t exist, but we can use a custom parser function provided by [forj-oss/maestro]::cacerts called cacerts_getkey.

Sample Code to create in forj-config:

```
[forj-config] / modules / custom_certs / manifest / review_sslparams.pp

class custom_certs::review_sslparams
{
  $ca_certs_db = hiera('cacerts::ca_certs_db','/opt/config/cacerts'),
}

$ssl_cert_file_contents = cacerts_getkey( "${ca_certs_db}/custom/review.yourdomain.com.crt" )
$ssl_key_file_contents = cacerts_getkey( "${ca_certs_db}/custom/review.yourdomain.com.key" )
$ssl_chain_file_contents = cacerts_getkey( "${ca_certs_db}/custom/chain.crt" )
if $ssl_cert_file_contents != ''
{
  notify('custom cert found for ci ssl_cert_file_contents':
    before => Class['cdk_project::gerrit'], # here we call the class we want to run before.
  )
} else
{
  warning('ci ssl_cert_file_contents is empty for custom cert')
}
```

• The default location for certs will be in `/opt/config/cacerts` for maestro node. Under this folder we will place the custom certs in a custom folder. The notify resource will be used to sequence the cert configuration of the parameters prior to any actions on the classes from gerrit.

• Note, the chain.crt is simply the collection of pem files for the certificate signers public keys that are created for the trust authority. You can make this empty if you are using a well known provider that comes installed with the browsers you want to support.

• repeat the steps for the ci node, by changing all the `review` names to `ci`.

• **Note, for the ci node, change**: `before => Class['cdk_project::gerrit'],`  
  `to before => Class['cdk_project::jenkins']`,

4. Un-comment or add this sections to maestro.yaml in [forj-config]::runtime_project This will place all the files from your custom_certs module into `/opt/config/cacerts/custom` on the maestro node.

Sample Code to create in forj-config:

```
[forj-config] / modules / runtime_project / files / hiera / layouts / maestro.yaml
```
5. Setup the master password for certs that were created in step 1 on the maestro box. The password itself will be encrypted on maestro under `/etc/puppet/hieradata/common.eyaml` with public/private keys found in `/etc/puppet/secure`. Make sure `thepassword` matches the password you selected in step 1 for `secretpass`.

**ssh to maestro node:**

```
$ sudo -i
$ cd /etc/puppet/secure
$ eyaml encrypt -l 'cacerts::custom::ca_pass' -s 'thepassword' | grep "cacerts::custom::ca_pass: ENC" >> /etc/puppet/hieradata/common.eyaml
```

6. Add the following yaml to `[forj-config]::runtime_project / files / hiera / layouts / <node>.yaml` files. If you are setting up custom certs for `review` and `ci` nodes, then you will update the following two files:

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>/modules/runtime_project/files/hiera/layouts</td>
<td>review.yaml</td>
</tr>
<tr>
<td>/modules/runtime_project/files/hiera/layouts</td>
<td>ci.yaml</td>
</tr>
</tbody>
</table>

- Add another class to classes: array.
- Set the `ssl_*_file_contents` section to the class param values.

**review.yaml contents:**

```yaml
classes:
  - custom_certs::review_sslparams

# these will be automatically created if we pass them in empty.
cdk_project::gerrit::ssl_cert_file_contents: "{%::custom_certs::review_sslparams::ssl_cert_file_contents%}"
cdk_project::gerrit::ssl_key_file_contents: "{%::custom_certs::review_sslparams::ssl_key_file_contents%}"
cdk_project::gerrit::ssl_chain_file_contents: "{%::custom_certs::review_sslparams::ssl_chain_file_contents%}"
```

**ci.yaml contents:**

```yaml
classes:
  - custom_certs::ci_sslparams

# these will be automatically created if we pass them in empty.
cdk_project::jenkins::ssl_cert_file_contents: "{%::custom_certs::ci_sslparams::ssl_cert_file_contents%}"
cdk_project::jenkins::ssl_key_file_contents: "{%::custom_certs::ci_sslparams::ssl_key_file_contents%}"
cdk_project::jenkins::ssl_chain_file_contents: "{%::custom_certs::ci_sslparams::ssl_chain_file_contents%}"
```

7. Execute puppet apply commands as root on **maestro, ci, and review** nodes in that order. Connect to the maestro node and run these commands:
$ sudo -i
$ puppet agent -t
$ salt -E '(ci|review).*' cmd.run "/usr/bin/puppet agent -t"

8. If certs don’t imediately install, you can also restart apache services on each node:
$ salt -E '(ci|review).*' cmd.run "service apache2 restart"

**FAQ**

... How do I create a new project?

Creating a new project on a redstone forge means creating a new Gerrit repository. We use the CI workflow of the forge itself to manage the project creation process. Configuration files are modified and updated to provide the administrator of the forge an opportunity to review the commit. Currently we do not provide automatic review option, but one could be setup using zuul gate triggers.

... re-trigger the verification for project create change request?

If your forge did not trigger a verification check for the project creation request, it is possible to re-trigger the request on the change request. Go to the change request and add a new comment. Make the comment text say: ‘recheck no bug’. This should trigger a zuul gate check for the change request.

... approve a new project creation request on gerrit?

First you must be the administrator of your forge or contact and the administrator of the forge you will try to access. The approving user must be added to the group, forj-core. This can be done in Gerrit from the Admin->Groups menu by the Gerrit administrator. Once done, the user added can then administer approvals by adding +2 for Code Review and +1 for Approved on the change request.

... change the group that approves changes for forj-config on gerrit?

Approval permissions for groups is managed by the forj-config acl’s file. This can be updated with a change request update to the forj-config source on the file : [forj-config.git/modules/runtime_project/files/gerrit/acls/production/forj-config.config

Change the group forj-core to a new group name. If the group does not exist a new one will be created.

### 1.1.5 Tuleap

Tuleap is the first Libre suite for Enterprise Application Lifecycle Management (ALM). Tuleap is used by fortune 500 companies & innovative start-ups to make software projects more productive, collaborative & industrialized whatever the processes

Tuleap is technology and process agnostic. The tool should adapt to the user is our motto.
Tools and features

- **Easy administration**
  - Quickly create ready-to-use projects & configure tools by project
- **Project Management**
  - Agile, V-model & custom processes managed with one single platform
- **Tracking**
  - Per project trackers with extensive customization & permissions options
- **Source code**
  - Integration with leading tools: Git, SVN
- **Document Management**
  - Create & version documents & wiki pages

Full description of features on [http://tuleap.net](http://tuleap.net)

Documentation

Tuleap project provides a comprehensive user guide.

Tutorial

Tuleap comes with a ready to work project for experimentation, it takes 5 minutes to familiarize yourself with the key component of the product.

During the tutorial you will learn how to use the agile dashboard, tuleap trackers and git.

Getting started

First, open %url%, click login as log as “richard_cover” with “welcome0” as password.

You land on your personal page, made of widgets. All those widgets are configurable, you can move them around, remove those you don’t need, add some others (see customize button).

On top left widget, you have the list of your projects, click on “Guinea Pig” to enters one.

**Guinea Pig project**

A project expose a list of services, those services are on the left part of the interface (the sidebar). This sidebar can be collapsed to save space with the little arrow at the bottom.

For this tutorial, we will do some actions on 2 of those services:

- Agile Dashboard
- Git

We will pick up a task on the agile dashboard and commit some stuff in git
Agile Dashboard

Open Agile Dashboard

You can see that there is a Sprint on going (sprint 1) with 2 elements open. On the left, you see the release the sprint belongs to.

On the top, you can access releases and sprints that were done and planned (What’s next)

Click on “Cardwall” for Sprint 1

On the cardwall there is a story and a task that are waiting to be done.

Drag the “Add Readme” task from the “to be done” column and drop it into the “On Going” column. Click on the small dash (-) next to “Assigned to” and assign the task to yourself.

Now we will push some code in git for this task

Git

Go on git service. There is nothing yet so create a new repository by entering “gpig” in the text field and click create.

The creation of the repository takes usually 1 or 2 minutes, wait for it and reload the page

When the repository is created you can use it, let’s clone on your workstation:

```bash
$> git clone http://URL/git/gpig8/gpig.git
```

Now create a new file “README” with some content into “gpig” directory and commit

```bash
$> cd gpig
$> $EDITOR README.txt
$> git add README
$> git commit -m "task #1 Add readme"
$> git push
```

Please note the commit line, the reference to the task from the agile dashboard

Now go back on the web site, on the git repository viewer, you can see the commit you just pushed.

Click on the commit message, you access to all the details of the commit. Here Tuleap recognized things for you, the “task #1” was detected as pattern to a element in your project and automatically turned in to a link.

Click on the link, you end-up on the task artifact. Change the status to “Done”

Note: you can see the backlink to the git commit.

Agile dashboard 2nd

Go back on the Cardwall for Release 1.0 > Sprint 1

You can see that the card is now in “Done” column.

Congratulations you completed a first round of contribution !

Going further

You got the key principles, know you can:

- Create your own account (Register user on the top right)
• Create a new project (In project drop down menu)
• Start hacking

What you can do next:
• Publish documents in “Docman” and get it reviewed by team mates thanks to informal “Approval tables” (Docman > Approval tables)
• Customize the User story format to add your own (Tracker > Administration > Manage fields)
• Create a bug tracker and set a custom workflow on it (Tracker > Administration > Workflow)
• Restrict git access to only project members and set email notifications (Git > Settings > Permissions)
• ... and share!

1.1.6 Getting help

Forj is a community project. Check the community page to get in touch with other Forj’ers who may be able to help.

1.1.7 Glossary

**Blueprint**  Source code which tells Maestro how to install, integrate, operate, update, backup, restore, relocate a forge. A blueprint describes a forge programatically.

**Catalog**  The list of blueprint available. The public blueprints are hosted on Forj’s forge.

**Forge**  The set of tools - installed by Maestro - by parsing the blueprint. This is the end product, and what the developers will actually use. This process is also used when restoring data to a new forge (safe restore).

**Forj’s forge**  Where Forj itself is developed. review.forj.io. Most of the code hosted there is also published on Github automatically.

**forj-config**  Special GIT project hosted on a forge which holds the configuration of the forge.

**Maestro**  Maestro manages all the tools from a forge. For forge users, it provides a web based user interface easily access all the tools that are used by a project. You also go to Maestro UI to register, so that an administrator can later provision your privileges. Forge administrators can manage projects and users and administrate the forge (status, backup/restore, take home).

**Portal**  Available at portal.forj.io, the portal is a mean to browse the catalog of forges and provision forges in few clicks. As an enterprise, you can replicate such portal and offer “forges as a service”. Forj’s portal uses HP CSA product.

**Take home, Relocation, Relocate**  Process initiated by Maestro to move all the systems of a forge from one location (public cloud, private cloud or on premises systems) to another.

1.1.8 License

Generally, Forj is licensed under the Apache v2 license. Please check the source code itself for specifics.

1.1.9 aPaas

Application platform as a service (aPaaS) is a cloud service that offers development and deployment environments for application services.

The following blueprints:
- redstone (aPaas details for this blueprint can be found under: redstone/puppet/modules/runtime_project/files/apaas)

include a CI/CD node with the CLIs necessary to interact with the following aPaas environments:
- Hewlett-Packard Helion Development Environment
- ActiveState Stackato
- Any vendor that supports CloudFoundry

The overall aPaas integration scenario is for a blueprint to include a node (OS depends on the aPaas vendor’s CLI OS support - almost all run on Linux/Mac OSX) that will execute the CLIs to interact with their environment. More support for aPaas vendors can be easily added by simply installing on the CI/CD node the required CLIs via the blueprint’s orchestration mechanism.

### 1.2 Developer guide

#### 1.2.1 Develop for Forj

Forj is an Open Source project. It couldn’t exist without contributions from the developer community. This is the place to start!

#### 1.2.2 Contribute

**Governance Model**

**Organization**

The Forj project is an open community of developers, and the code is governed by the Apache Software License v2.0. The community is comprised of Forj development team members, contributors from HP, and external contributors. Since the community structure is evolving, in this initial stage, all components will have oversight by the Forj core team, described below.

**Core team**

The Forj core team is composed by the following individuals:

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Nickname (IRC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Management</td>
<td>Rafael Garcia</td>
<td>RafGar</td>
</tr>
<tr>
<td></td>
<td>Olivier Jacques</td>
<td>osanchez</td>
</tr>
<tr>
<td>Product Owner</td>
<td>Edward Raigosa</td>
<td>wenlock</td>
</tr>
<tr>
<td>Overall Architect</td>
<td>Kathryn Kuhn</td>
<td>kuhnk</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>Miguel Quintero</td>
<td>miqui</td>
</tr>
<tr>
<td>Lead</td>
<td>Christophe Larsonneur</td>
<td>chrissss</td>
</tr>
<tr>
<td>Lead</td>
<td>Oscar Romero</td>
<td>homeless</td>
</tr>
<tr>
<td>Lead</td>
<td>Arne Luehrs</td>
<td>che-arne</td>
</tr>
</tbody>
</table>

We encourage participation by anyone who wants to collaborate on the direction of the Forj Project. The best way to get involved is to join the community and take an active role as a contributor. As the community grows, and public developers take on more leadership in contributing to components, we hope to extend maintainer roles to external contributors from the Forj community at large.
Communication and Process

The Project uses well known tools and processes to make contributing and communicating straightforward.

These resources include this website, our own review system based on Gerrit, GitHub to publish code, a bug tracker, mailing list and an IRC channel.

A mechanism inspired by the well known Linux “Developer Certificate of Origin (DCO)” is used to sign off contributions.

Everyone contributing to the Forj Project or participating on the IRC, mailing list or other resources provided by the Project must abide by the Community Guidelines. While creative ideas and respectful debates on topics are encouraged, the Forj Project reserves the right to act as needed to protect the integrity of the Project, including the removal of any posts deemed inappropriate or offensive.

Community members should note that the information submitted via the IRC, mailing list or any other communications, resources of the Forj Project will immediately become public information.

How do I contribute?

1. Join the Community: Joining allows you to contribute code, participate in IRC and mailing list, report and track bugs and receive updates on the latest Forj developments.

2. Decide Your Purpose: Do you want to add a feature to Forj, contribute to a blueprint or enhance the documentation?

3. Get Familiar with the Components: Once you determined your activity, check out our repos on Forj’s Gerrit or GitHub to review our components and READ ME files.

4. Discuss Your Idea: Before you even start updating components, you should post your ideas on the IRC or mailing list. You can find out if someone else is working on a similar idea or community members can help you refine your concept.

5. Review the Governance Model: Follow our process for how to get your contribution accepted as detailed under our Governance Model.

How We Accept Contributions

The Contribution Process

- Take a look at our issue list and consider submitting a patch. You may have a look at our “fruits” issues (from the low hanging kind) to get started.

- For big changes, the contributor communicates with the Project via mailing lists or IRC or Issue List to get feedback before submitting code

- Contributor agrees to the dco - see the link for format and meaning

- Contributor adds necessary Apache v2.0 header for each new file contributed
Contributing code

Forj’s code development process mimics the one used by the OpenStack project. If you are an OpenStack contributor, you will feel at home.

For more information on this process, please refer to Openstack’s Gerrit workflow.

Useful links:
Contributing documentation

Documentation is also an open source project. The sources are available from the main Forj repository on Github. We encourage you to make improvements, whether big or small, to this documentation.

Useful links:

- Forj’s documentation repository
- Sphinx document generator

Forj’s user and admin documentation (http://docs.forj.io/) is hosted on “readthedocs.org” (http://readthedocs.readthedocs.org/) at this address: https://readthedocs.org/projects/forj/ The documentation uses “RST” markup language and is generated with Sphinx http://sphinx-doc.org/

To contribute to the documentation

- Fork the docs repository from github: http://github.com/forj-oss/docs
- Setup your sphinx environment on your PC: http://sphinx-doc.org/
- Edit the doc
• test it with “make html”
• Open a pull request on github and interact with the core team
• Once reviewed, the changes are merged by the core team in the Github repository
• The documentation gets refreshed automatically thanks to a web hook

Criteria before submitting a contribution

• Contributor has verified that their changes do not break any of the builds
• Contributor has provided or updated unit tests, if there is an existing unit test structure for any of the components affected
• If there is no unit test structure, the contributor has thoroughly tested their changes manually, and can describe the results
• Code is in the style of the code that surrounds it

During Review the Maintainer Will

• Look to see that you have the DCO on file and added the necessary Apache v2.0 license headers to the files (automated)
• If you are someone entirely new to the Project, they may get in touch with you via the contact information you have provided
• If there are anomalies such as inconsistent name or email address between signoffs, they may ask you to clarify
• This process may take some time, since we may conduct testing, and there may be concurrent activity which must be checked for merge conflicts, architectural issues, etc.

After the merge

• Your commits will be merged onto the master branch
• The maintainer’s identity who accepted your pull request will be recorded in the merge
• Congrats, you are now a Forj contributor!

Community guidelines

We want to keep the Forj Community an interactive, open place to participate, but we need your help. We expect participants across all areas of the website - including the mailing list, forum, bug tracker - will:
• Be courteous and polite to fellow community members
• Respect other people: no racial, gender or other abuse will be tolerated
• Not swear
• Ensure that you have legal rights for posting your content and are not violating any copyright, trade secrets, trademark or other proprietary rights.
• Make sure your posts are sent to the appropriate channels and are relevant to the discussion. You can visit the mailing list page to determine the right place to post your question or discussion topic.
1.2.3 Guide to the Developer Grant and Certificate of Origin

Things To Consider Before Contributing to the Project

To help with understanding what’s required to contribute to Forj, we are providing guidance on why the Project has made certain decisions.

- The Project has selected the Apache 2.0 license, popular for its permissive and flexible properties, while still encouraging a collaborative community.

- The Project has selected an approach to contributions inspired by the popular Linux Developer “Certificate of Origin”. This approach makes it simple for new contributors to get started, and avoids bureaucracy in tracking contributions and contributors.

- To indicate accordance, each contributor must validate accordance on Forj’s Gerrit:
  - Go to Forj’s Gerrit then to Settings / Agreements / New Contributor Agreement / DCO
  - Review carefully the agreement, if you agree, enter “I AGREE” in the box, and click “Submit agreement”

Agreeing to the DCO means that you state that the following are true of your contribution:

- I created this contribution/change and have the right to submit it to the Project; or

- I created this contribution/change based on a previous work with a compatible open source license; or

- This contribution/change has been provided to me by someone who did (a) or (b) and I am submitting the contribution unchanged.

- I understand this contribution is public and may be redistributed as open source software.

- I understand that I retain copyright ownership in this contribution and I am granting the Project a copyright license to use, modify and distribute my contribution. The Project may relicense my contribution under other OSI-approved licenses.

Note: You must confirm the nature of your contribution with the actual certificate and license language.

Why are we including the final paragraph which is not part of the Linux DCO?

Under the Apache Software License v2.0, each contributor provides a copyright license for their contribution to all licensees of the Project. The Project includes some source code files, which originate under other OSI-approved licenses compatible with the Apache license. Modifications to such files will typically be licensed by the Project under the original open source software license.

In some cases the Project may need to relicense your contribution under other OSI-approved open source licenses in order to maintain the Apache-based licensing. For example, if your contribution is affected by GPL v2 code, the Project may dual-license your code under both Apache 2.0 and GPL v2 or LGPLv2.1. This satisfies the GPL requirements, while still allowing other users to make use of the code under Apache.

The Certificate

Forj Developer Grant and Certificate of Origin 1.0 By making a contribution to the Forj Project (“Project”), I represent and warrant that:

1. The contribution was created in whole or in part by me and I have the right to submit the contribution on my own behalf or on behalf of a third party who has authorized me to submit this contribution to the Project; or
2. The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right and authorization to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license) that I have identified in the contribution; or

3. The contribution was provided directly to me by some other person who represented and warranted (a) or (b) and I have not modified it.

4. I understand and agree that this Project and the contribution are publicly known and that a record of the contribution (including all personal information I submit with it, including my sign-off record) is maintained indefinitely and may be redistributed consistent with this Project or the open source license(s) involved.

I hereby grant to the Project, HP and recipients of software distributed by the Project a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, modify, prepare derivative works of, publicly display, publicly perform, sublicense, and distribute this contribution and such modifications and derivative works consistent with this Project, the open source license indicated in the previous work or other appropriate open source license specified by the Project and approved by the Open Source Initiative (OSI) at http://www.opensource.org.

1.2.4 Architecture

High level architecture

Forj Blueprint catalog is hosted by HP, leveraging “HP CSA - Cloud Services Automation”. If you are an enterprise, this allows you to run your own HP CSA and expose CI/CD stacks as a service for your R&D teams. You do not have to run HP CSA to create a forge. You can use the “forj command line” to create CI/CD environments.
Detailed architecture

1.2.5 Gardener

Module in charge of managing configurations, credentials, user-data for forj forges, as well as creating and destroying the static nodes of the forge.

Install Gardener

The installation of gardener.

1. During the creation of master, in the file 10-puppet.pp the instruction `puppet apply $PUPPET_FLAGS --modulepath=$PUPPET_MODULES /opt/config/production/puppet/manifests/bootstrap_hiera.pp` in the Stage [Main] the file init.pp from gardener is loaded.

2. The file init.pp include the class requirements, this loads the file requirements.pp.

3. The file requirements.pp, look for the requirements and their versions.

Figure below shows the flow of control.

Verifying Installation and Loading Plug-ins

At the end of 10-puppet.pp, one execute the command `puppet agent $PUPPET_FLAGS --waitforcert` which verifies the install and loads the puppet plug-ins (the gardener’s plug-ins are also loaded).

At the end of the file 20-instantiate-forj-at-boot.sh, one execute the command `puppet agent --debug --verbose $PUPPET_FLAGS --waitforcert`, which verifies the install and checks the status of the nodes, if any node is listed as not found or not created, it is created.

Loading Provider and Credentials

When the plug-ins are loaded, will load the provider and credentials configuration

1.2. Developer guide
Install Gardener
10-puppet.sh / 20-instantiate-forj-at-boot.sh

- runs puppet agent load the gardener plugins each time that loader.rb or a .rb file that includes loader is loaded. search for FOG_RC configuration inside cloud.fog file
- once the nodes have been created also check for the nodes status incase not found create them
1. Load `pinas/loader.rb`.
2. Loader will try to get the provider using `provider.rb`.
3. Provider.rb will search for the configuration section called FOG_RC inside the configuration file `cloud.fog`.
4. Next `compute_public_ip_lookup.rb` will be loaded.
5. This one will check for if the fog_credentials exist if no will create with `fog_credentials.rb`.
6. fog_credentials will add the feature credentials to yaml and try to load it from the file `cloud.fog`.
7. In case that not found the `compute_public_ip_lookup.rb` will try to load FOG_RC using `pinas/loader.rb`.
8. Load `pinasdns/loader.rb` and will do 2 and 3.
9. load `domain_record_exists.rb` and will try to do 5 and 6 if necessary 7.

Creating User Data

20-instantiate-forj-at-boot.sh creates the nodes of the blueprint with user-data.

1. Is set a relationship `write-mime-multipart.template.py` before `boothook.template.sh`.
2. Is set a relationship `boothook.template.sh` before `boot-node.template.sh`.
3. Is set a relationship `boot-node.template.sh` before `cloud-config-node.template.yaml`.
6. Then the `gardener::srever_up.pp::gen_userdata` will be executed.
7. This will use `gardener::gen_userdata.pp` execute the files necessary for the generation.
8. `gardener::gen_userdata` will create the file `mime.txt` according to the relationships created before.
Creating Nodes

This is one of the main functionalities of gardener.

1. gardener::params.pp has as one of its attributes compute, which call to compute.rb.
2. compute.rb will call actions.rb method create, while have servers in the template this will be executed.
3. actions.rb get the server name using common.rb get_server_name function.
4. Next, actions.rb will check if the server already exists, using pinnascompute.rb server_exist? function.
5. This function uses common.rb find_match function to check if exists.
6. If not exist pinnascompute.rb sill use the method create_server.
7. This method gets parameters from mime.txt and try to create new server.

Destroying Nodes

this functionality will destroy the nodes created.

maestro::orchestrator::unwindallservers will use gardener::server_destroy.

1. This one will use compute.rb to destroy the servers using actions.rb.
2. To destroy the server will do it one by one according to the server template.
3. To do it first will get the server name using common.rb get_servername.
4. Now that it has the server name will use pinnascompute.rb server destroy method.
5. To destroy it first it recover the server using common.rb find_match.
6. And then it just destroy the server.

1.2.6 Development environment

Coming soon...
1.2.7 Forj FAQ

This is a list of Frequently Asked Questions about Forj. Feel free to suggest new entries!

... What is Forj license?

See License

... I want a forge from Forj, but I do not have a cloud

Amongst the possibilities, you could sign up for an HP Cloud account and use Forj’s portal to provision your forge there. Last time we checked, HP Cloud offered $300 credit over a period of 3 months. Enough to host a “Redstone” forge for free for 3 months. Sign up, get the credentials, and **install** a forge with Forj today.
CHAPTER 2

Indices and tables

• genindex
• search
B
Blueprint, 21

C
Catalog, 21

F
Forge, 21
Forj’s forge, 21
forj-config, 21

M
Maestro, 21

P
Portal, 21

R
Relocate, 21
Relocation, 21

T
Take home, 21