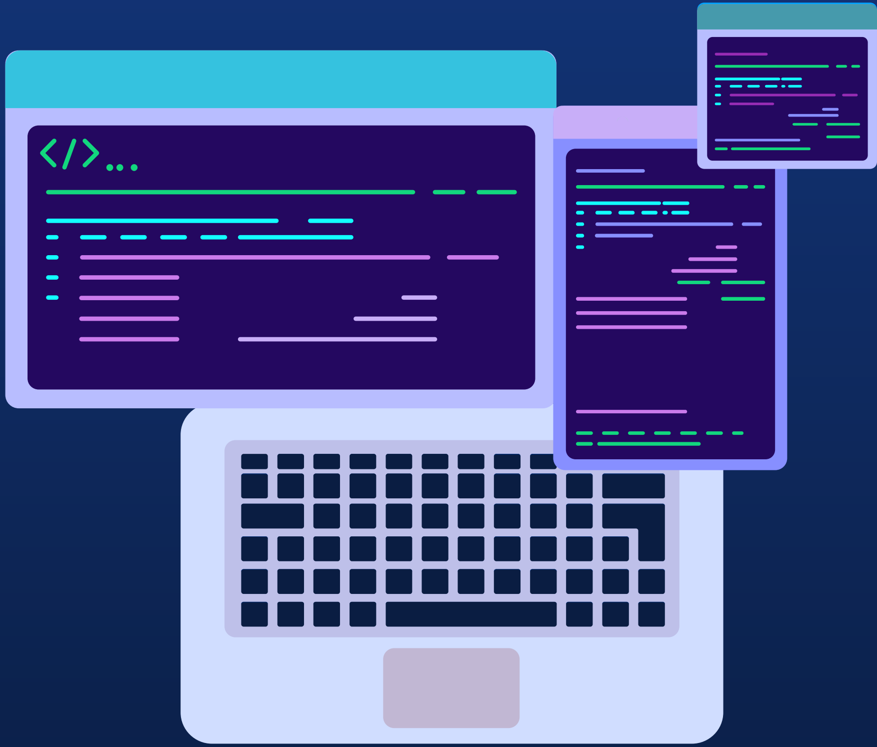
Managing Jenkins with Progress Chef - How and Why

ChefConf 2023



Overview

Evolving to Chef
Moving to AWS+IAC
Growing Pains
Rebuilding Jenkins with Chef/JCasC
Code Examples
Supporting Migration to Microservices
Future Steps
Lessons Learned

Mendy Baitelman

VP, Platform at Prodege

Mendel Douek

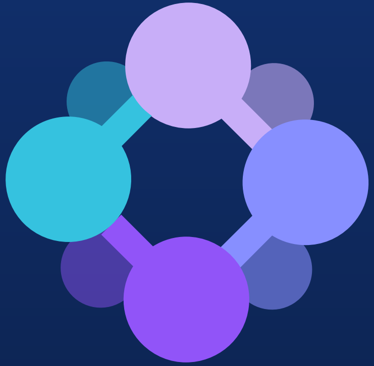
Jr. Platform Engineer at Prodege

prodege

Prodege is a marketing & consumer insights platform that uses its global audience of reward program members to power its business solutions. These members earn by searching, answering surveys, and shopping online and have redeemed over 2 billion dollars in rewards.

Evolving to Jenkins

Moving to AWS with Chef



Managing Chef with Jenkins

https://www.youtube.com/watch?v=8-HkLuS-f0s

Growing Pains

Rebuilding with JCasC

- Easy system updates
- Easy plugin updates
- Config as Code for everything [System/plugins/deps]
- Move user management to SAML
- Move secrets to secret store
- Ability to spin up test systems
- Support for multiple envs (split out QA/dev)

New Architecture

- Docker image on Ubuntu server
- Mounted volume for data
- AWS Network Load Balancer [NLB]
- AWS Parameter Store for secrets
- Terraform (for non Jenkins AWS resources)
- EC2, Network, SGs, IAM roles/policies (about 20/50)
- Chef *
- Jenkins Configuration as Code [JCasC]
- Declarative jobs only

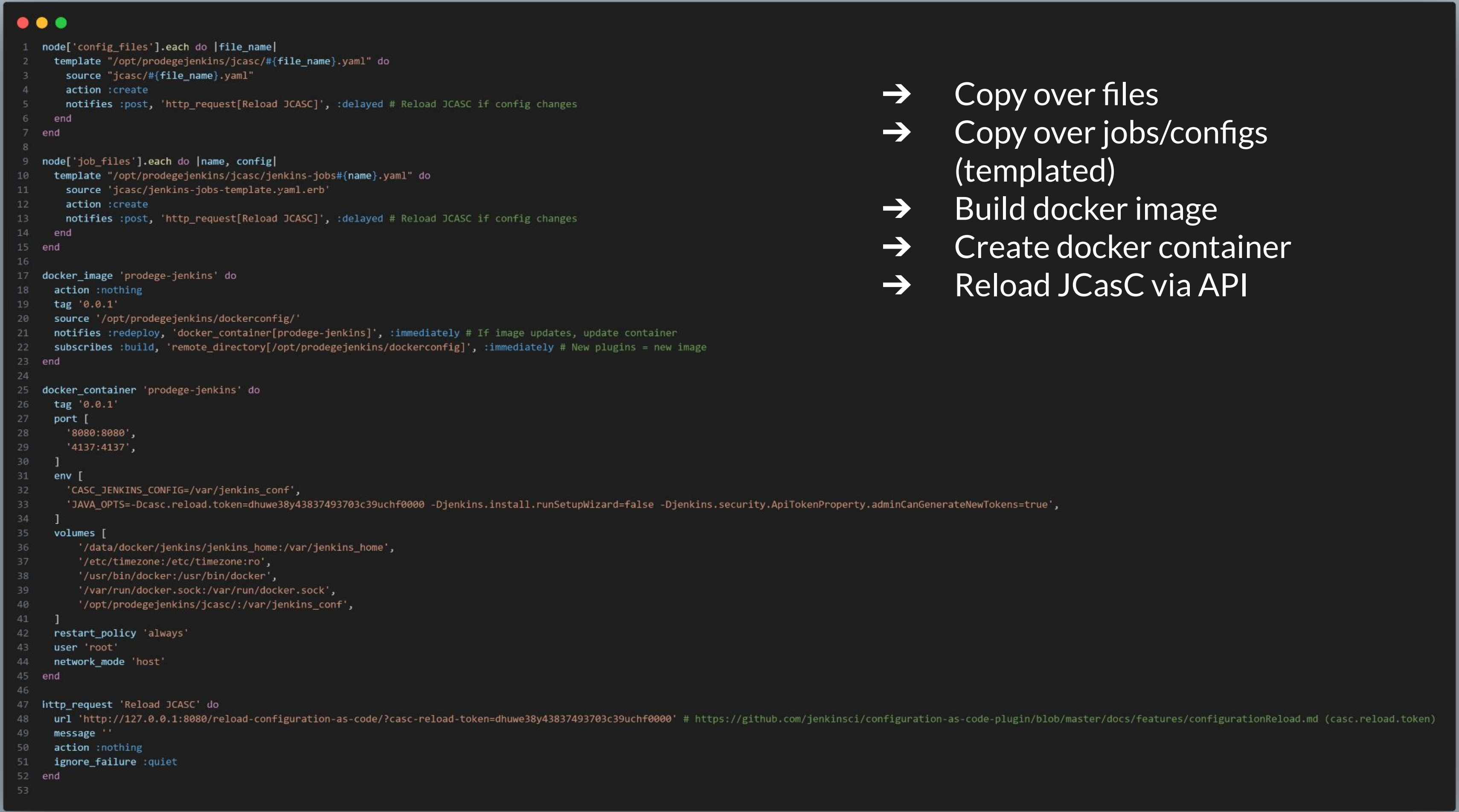
Just Add Chef

Why Chef?

- Support Multiple Environments
- Templating
- Pain-free System Updates
- Easier Jenkins Plugin Updates
- Test Kitchen
- Secret Store Use

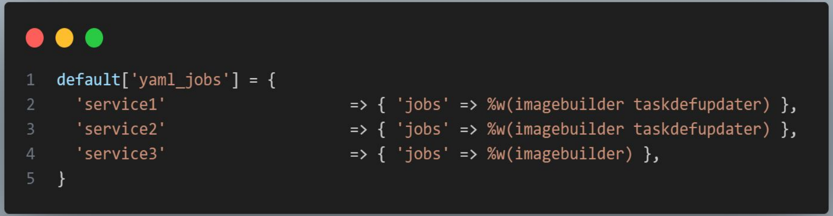
Code:

How it works

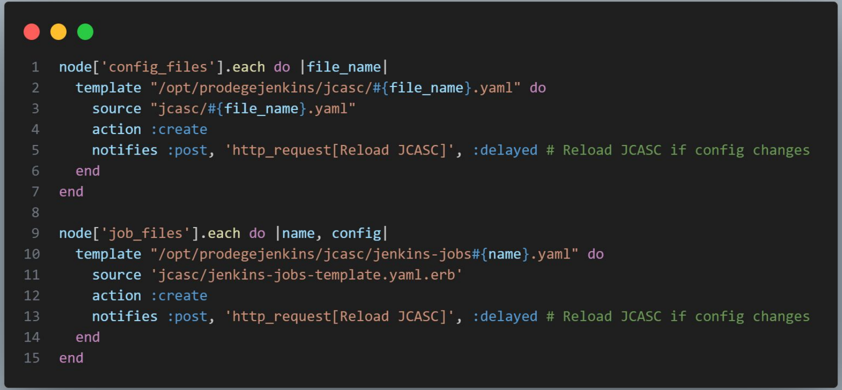


```ruby
node['config_files'].each do |file_name|
  template "/opt/prodegejenkins/jcasc/#{file_name}.yaml" do
    source "jcasc/#{file_name}.yaml"
    action :create
    notifies :post, 'http_request[Reload JCASC]', :delayed # Reload JCASC if config changes
  end
end

node['job_files'].each do |name, config|
  template "/opt/prodegejenkins/jcasc/jenkins-jobs#{name}.yaml" do
    source 'jcasc/jenkins-jobs-template.yaml.erb'
    action :create
    notifies :post, 'http_request[Reload JCASC]', :delayed # Reload JCASC if config changes
  end
end

docker_image 'prodege-jenkins' do
  action :nothing
  tag '0.0.1'
  source '/opt/prodegejenkins/dockerconfig/'
  notifies :redeploy, 'docker_container[prodege-jenkins]', :immediately # If image updates, update container
  subscribes :build, 'remote_directory[/opt/prodegejenkins/dockerconfig]', :immediately # New plugins = new image
end

docker_container 'prodege-jenkins' do
  tag '0.0.1'
  port [
    '8080:8080',
    '4137:4137',
  ]
  env [
    'CASC_JENKINS_CONFIG=/var/jenkins_conf',
    'JAVA_OPTS=-Dcasc.reload.token=dhuwe38y43837493703c39uchf0000 -Djenkins.install.runSetupWizard=false -Djenkins.security.ApiTokenProperty.adminCanGenerateNewTokens=true',
  ]
  volumes [
      '/data/docker/jenkins/jenkins_home:/var/jenkins_home',
      '/etc/timezone:/etc/timezone:ro',
      '/usr/bin/docker:/usr/bin/docker',
      '/var/run/docker.sock:/var/run/docker.sock',
      '/opt/prodegejenkins/jcasc/:/var/jenkins_conf',
  ]
  restart_policy 'always'
  user 'root'
  network_mode 'host'
end

http_request 'Reload JCASC' do
  url 'http://127.0.0.1:8080/reload-configuration-as-code/?casc-reload-token=dhuwe38y43837493703c39uchf0000' # https://github.com/jenkinsci/configuration-as-code-plugin/blob/master/docs/features/configurationReload.md (casc.reload.token)
  message ''
  action :nothing
  ignore_failure :quiet
end
```

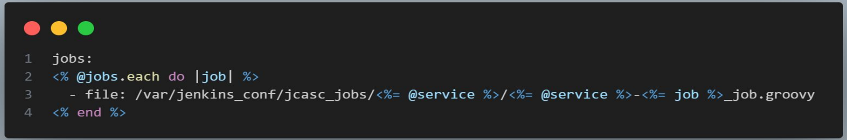
- ➔ Copy over files
- ➔ Copy over jobs/configs (templated)
- ➔ Build docker image
- ➔ Create docker container
- ➔ Reload JCasC via API



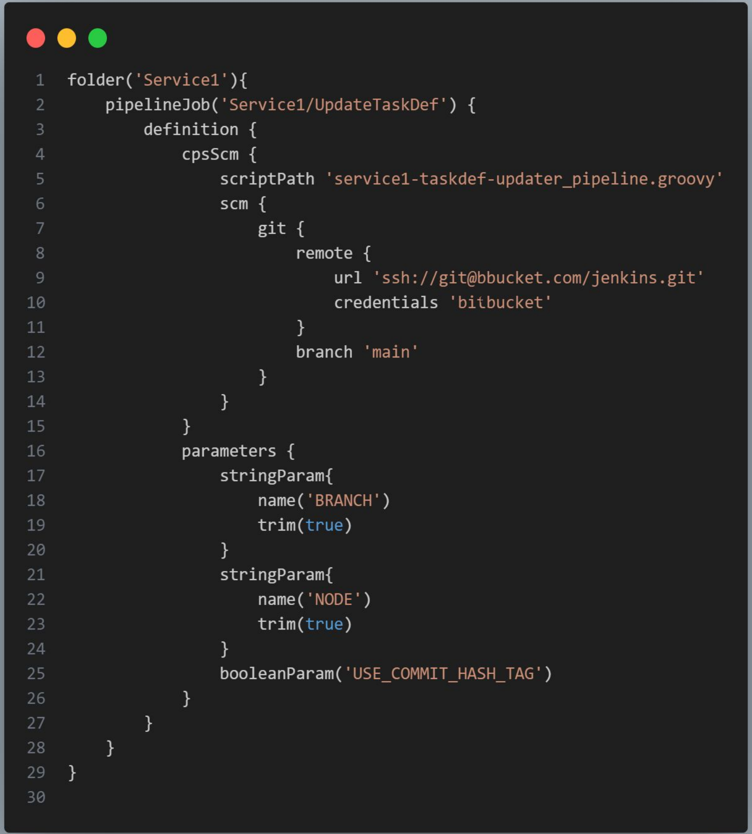
```
default['yaml_jobs'] = {
  'service1'                    => { 'jobs' => %w(imagebuilder taskdefupdater) },
  'service2'                    => { 'jobs' => %w(imagebuilder taskdefupdater) },
  'service3'                    => { 'jobs' => %w(imagebuilder) },
}
```



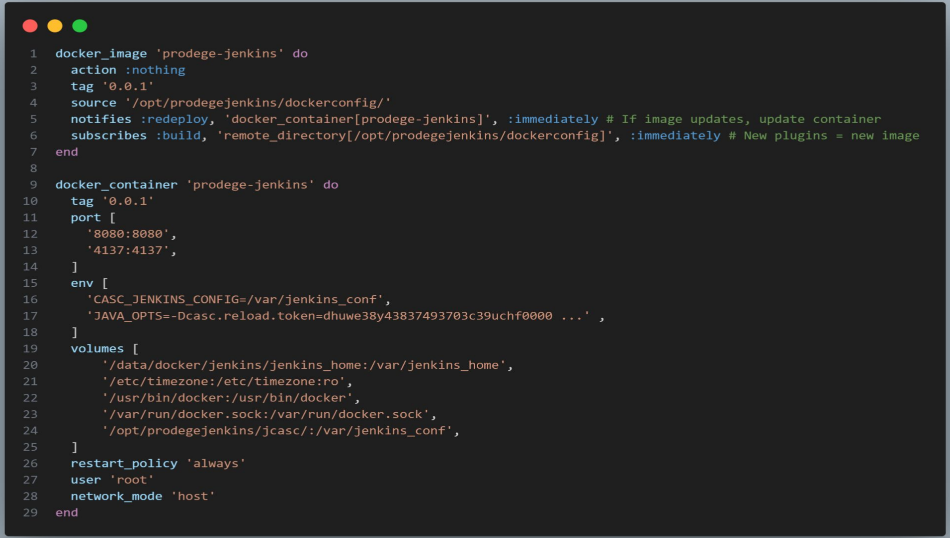
```
node['config_files'].each do |file_name|
  template "/opt/prodegejenkins/jcasc/#{file_name}.yaml" do
    source "jcasc/#{file_name}.yaml"
    action :create
    notifies :post, 'http_request[Reload JCASC]', :delayed # Reload JCASC if config changes
  end
end

node['job_files'].each do |name, config|
  template "/opt/prodegejenkins/jcasc/jenkins-jobs#{name}.yaml" do
    source 'jcasc/jenkins-jobs-template.yaml.erb'
    action :create
    notifies :post, 'http_request[Reload JCASC]', :delayed # Reload JCASC if config changes
  end
end
```



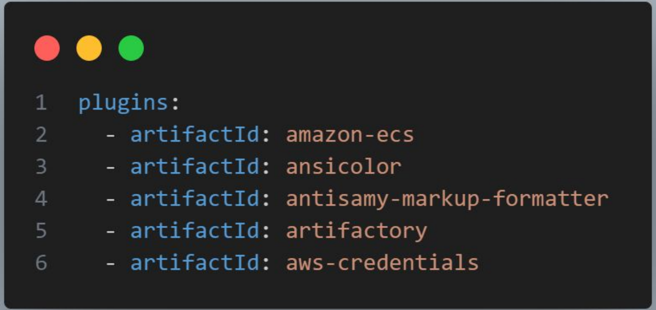
```
jobs:
<% @jobs.each do |job| %>
  - file: /var/jenkins_conf/jcasc_jobs/<%= @service %>/<%= @service %>-<%= job %>_job.groovy
<% end %>
```



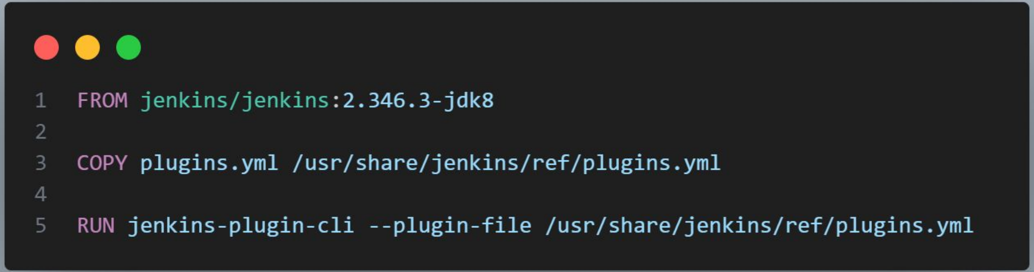
```
folder('Service1'){
    pipelineJob('Service1/UpdateTaskDef') {
        definition {
            cpsScm {
                scriptPath 'service1-taskdef-updater_pipeline.groovy'
                scm {
                    git {
                        remote {
                            url 'ssh://git@bbucket.com/jenkins.git'
                            credentials 'bitbucket'
                        }
                        branch 'main'
                    }
                }
            }
            parameters {
                stringParam{
                    name('BRANCH')
                    trim(true)
                }
                stringParam{
                    name('NODE')
                    trim(true)
                }
                booleanParam('USE_COMMIT_HASH_TAG')
            }
        }
    }
}

```



```ruby
docker_image 'prodege-jenkins' do
  action :nothing
  tag '0.0.1'
  source '/opt/prodegejenkins/dockerconfig/'
  notifies :redeploy, 'docker_container[prodege-jenkins]', :immediately # If image updates, update container
  subscribes :build, 'remote_directory[/opt/prodegejenkins/dockerconfig]', :immediately # New plugins = new image
end

docker_container 'prodege-jenkins' do
  tag '0.0.1'
  port [
    '8080:8080',
    '4137:4137',
  ]
  env [
    'CASC_JENKINS_CONFIG=/var/jenkins_conf',
    'JAVA_OPTS=-Dcasc.reload.token=dhuwe38y43837493703c39uchf0000 ...' ,
  ]
  volumes [
      '/data/docker/jenkins/jenkins_home:/var/jenkins_home',
      '/etc/timezone:/etc/timezone:ro',
      '/usr/bin/docker:/usr/bin/docker',
      '/var/run/docker.sock:/var/run/docker.sock',
      '/opt/prodegejenkins/jcasc/:/var/jenkins_conf',
  ]
  restart_policy 'always'
  user 'root'
  network_mode 'host'
end
```

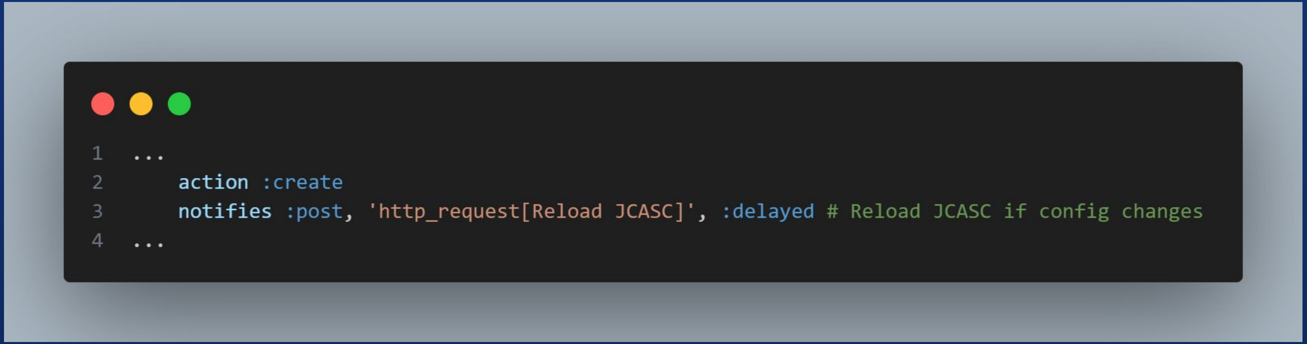


```yaml
plugins:
  - artifactId: amazon-ecs
  - artifactId: ansicolor
  - artifactId: antisamy-markup-formatter
  - artifactId: artifactory
  - artifactId: aws-credentials
```

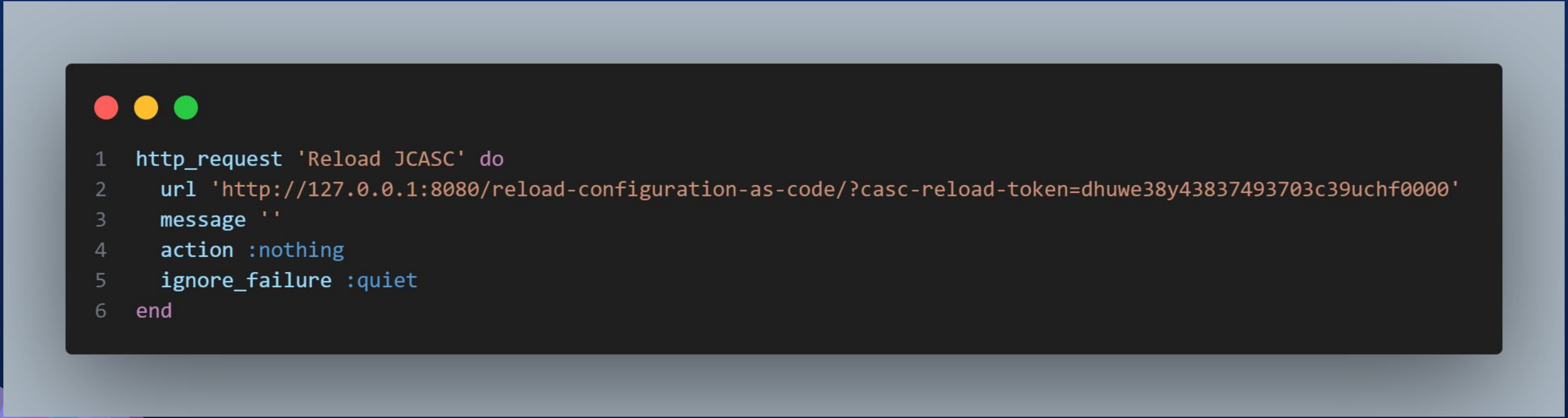


```dockerfile
FROM jenkins/jenkins:2.346.3-jdk8

COPY plugins.yml /usr/share/jenkins/ref/plugins.yml

RUN jenkins-plugin-cli --plugin-file /usr/share/jenkins/ref/plugins.yml
```

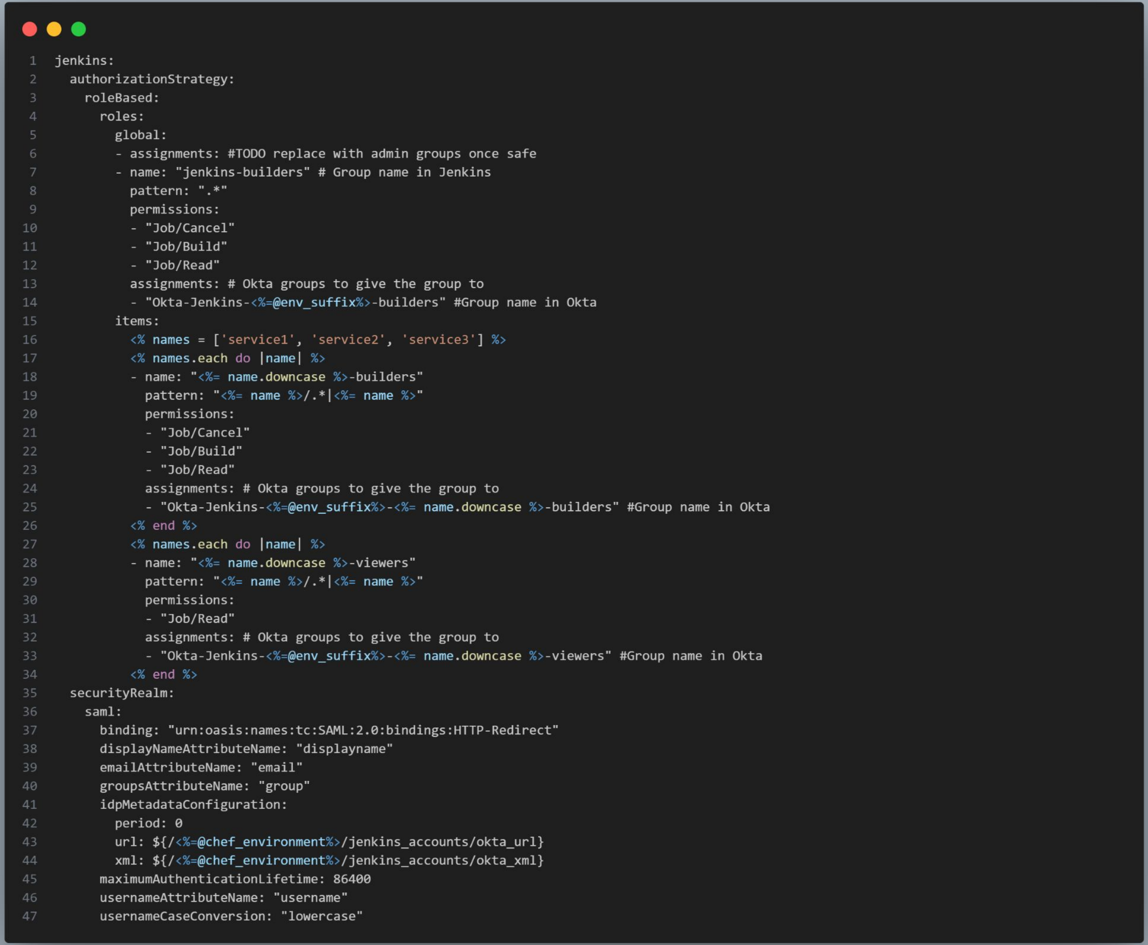
Hot Reloading



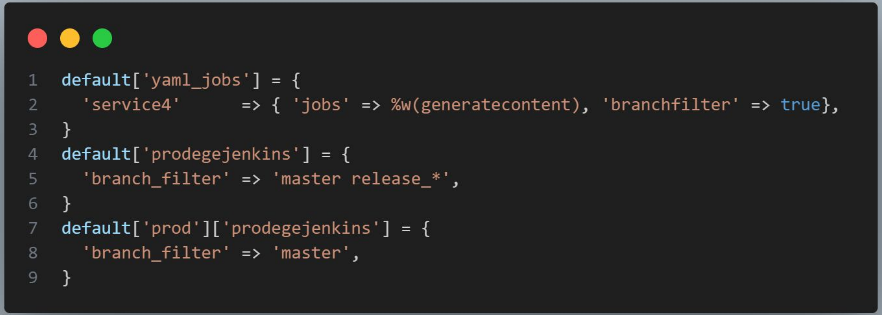
```
...
    action :create
    notifies :post, 'http_request[Reload JCASC]', :delayed # Reload JCASC if config changes
...
```



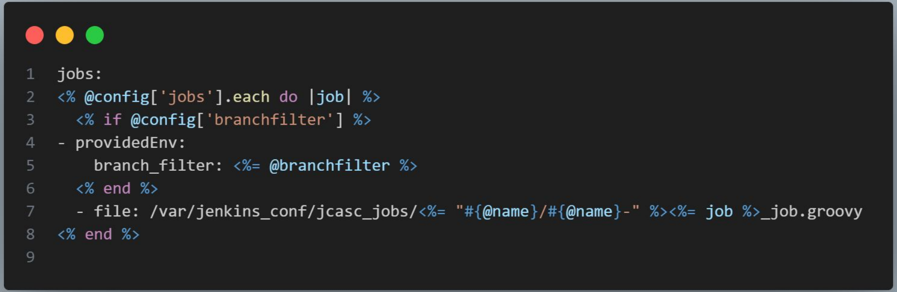
```
http_request 'Reload JCASC' do
  url 'http://127.0.0.1:8080/reload-configuration-as-code/?casc-reload-token=dhuwe38y43837493703c39uchf0000'
  message ''
  action :nothing
  ignore_failure :quiet
end
```



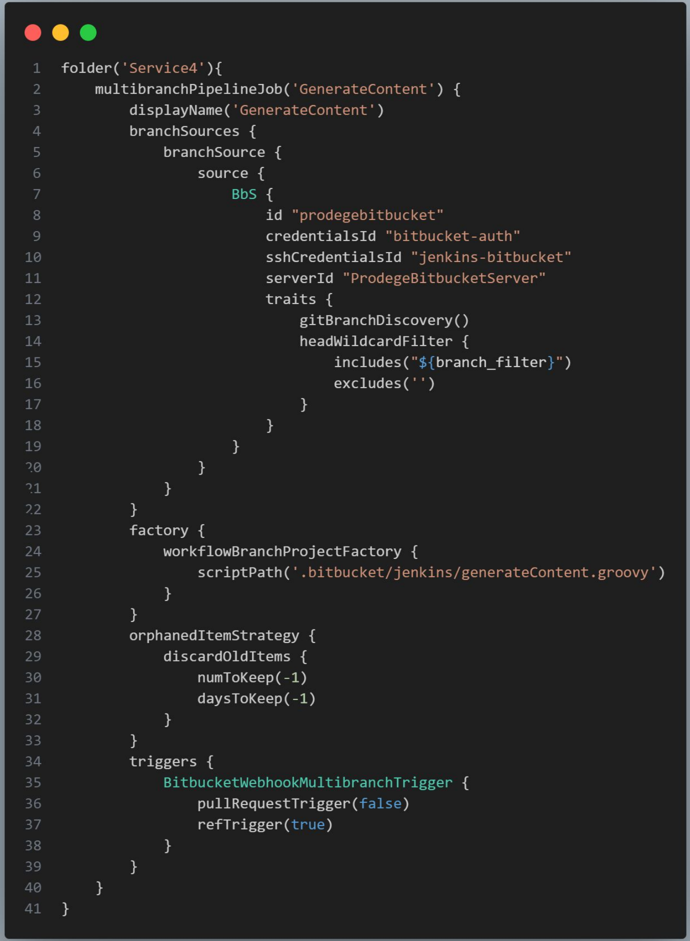
```
jenkins:
  authorizationStrategy:
    roleBased:
      roles:
        global:
        - assignments: #TODO replace with admin groups once safe
        - name: "jenkins-builders" # Group name in Jenkins
          pattern: ".*"
          permissions:
          - "Job/Cancel"
          - "Job/Build"
          - "Job/Read"
          assignments: # Okta groups to give the group to
          - "Okta-Jenkins-<%=@env_suffix%>-builders" #Group name in Okta
        items:
          <% names = ['service1', 'service2', 'service3'] %>
          <% names.each do |name| %>
          - name: "<%= name.downcase %>-builders"
            pattern: "<%= name %>/.*|<%= name %>"
            permissions:
            - "Job/Cancel"
            - "Job/Build"
            - "Job/Read"
            assignments: # Okta groups to give the group to
            - "Okta-Jenkins-<%=@env_suffix%>-<%= name.downcase %>-builders" #Group name in Okta
          <% end %>
          <% names.each do |name| %>
          - name: "<%= name.downcase %>-viewers"
            pattern: "<%= name %>/.*|<%= name %>"
            permissions:
            - "Job/Read"
            assignments: # Okta groups to give the group to
            - "Okta-Jenkins-<%=@env_suffix%>-<%= name.downcase %>-viewers" #Group name in Okta
          <% end %>
  securityRealm:
    saml:
      binding: "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
      displayNameAttributeName: "displayname"
      emailAttributeName: "email"
      groupsAttributeName: "group"
      idpMetadataConfiguration:
        period: 0
        url: ${/<%=@chef_environment%>/jenkins_accounts/okta_url}
        xml: ${/<%=@chef_environment%>/jenkins_accounts/okta_xml}
      maximumAuthenticationLifetime: 86400
      usernameAttributeName: "username"
      usernameCaseConversion: "lowercase"
```



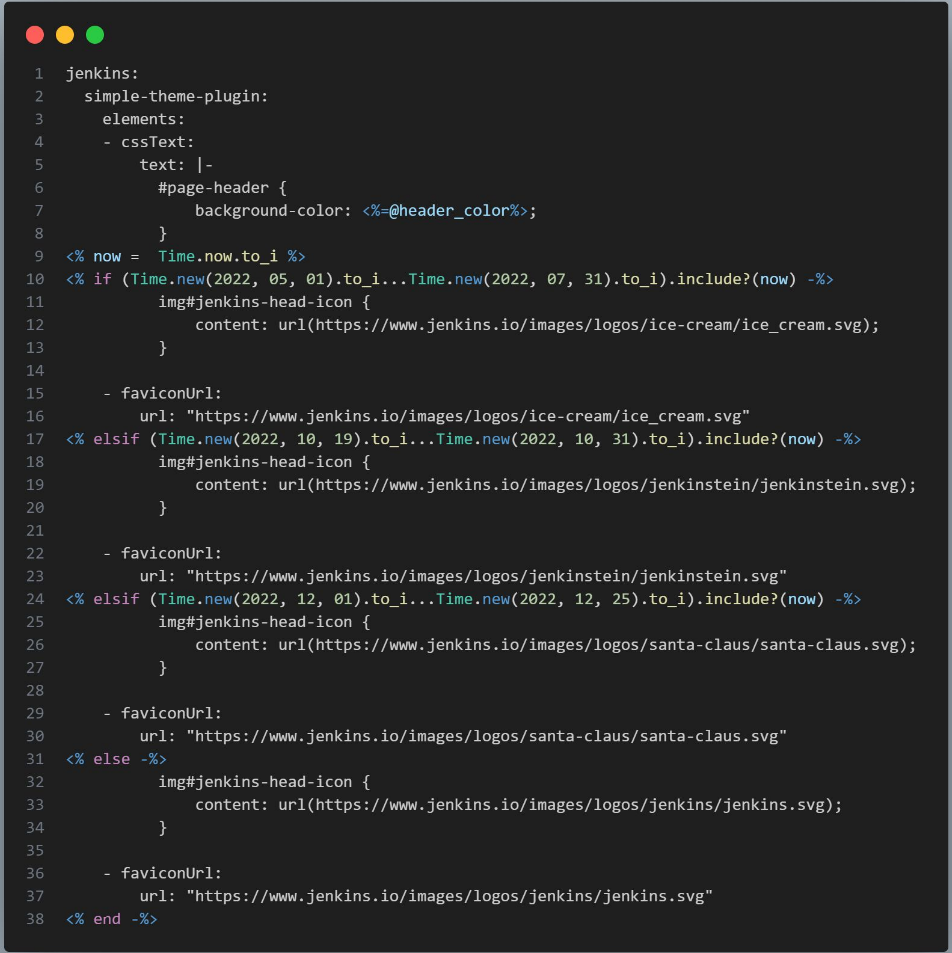
```
default['yaml_jobs'] = {
  'service4'      => { 'jobs' => %w(generatecontent), 'branchfilter' => true},
}
default['prodegejenkins'] = {
  'branch_filter' => 'master release_*',
}
default['prod']['prodegejenkins'] = {
  'branch_filter' => 'master',
}
```



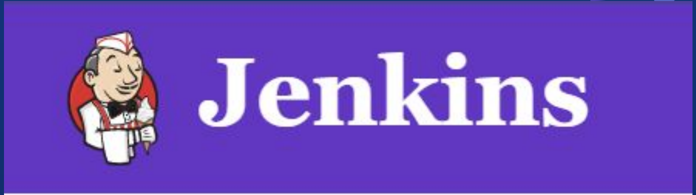
```
jobs:
<% @config['jobs'].each do |job| %>
  <% if @config['branchfilter'] %>
- providedEnv:
    branch_filter: <%= @branchfilter %>
  <% end %>
  - file: /var/jenkins_conf/jcasc_jobs/<%= "#{@name}/#{@name}-" %><%= job %>_job.groovy
<% end %>

```



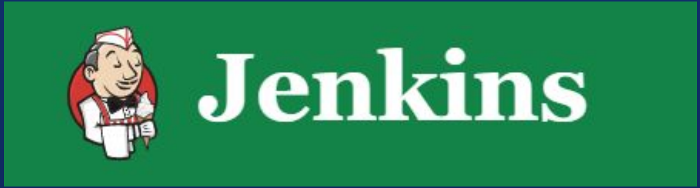
```
folder('Service4'){
    multibranchPipelineJob('GenerateContent') {
        displayName('GenerateContent')
        branchSources {
            branchSource {
                source {
                    BbS {
                        id "prodegebitbucket"
                        credentialsId "bitbucket-auth"
                        sshCredentialsId "jenkins-bitbucket"
                        serverId "ProdegeBitbucketServer"
                        traits {
                            gitBranchDiscovery()
                            headWildcardFilter {
                                includes("${branch_filter}")
                                excludes('')
                            }
                        }
                    }
                }
            }
        }
        factory {
            workflowBranchProjectFactory {
                scriptPath('.bitbucket/jenkins/generateContent.groovy')
            }
        }
        orphanedItemStrategy {
            discardOldItems {
                numToKeep(-1)
                daysToKeep(-1)
            }
        }
        triggers {
            BitbucketWebhookMultibranchTrigger {
                pullRequestTrigger(false)
                refTrigger(true)
            }
        }
    }
}
```



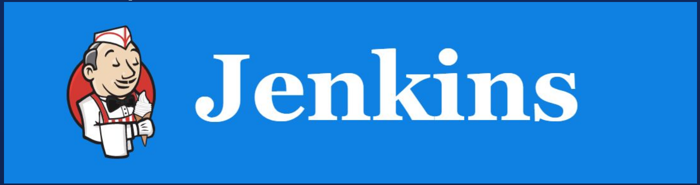
```
jenkins:
  simple-theme-plugin:
    elements:
    - cssText:
        text: |-
          #page-header {
              background-color: <%=@header_color%>;
          }
<% now =  Time.now.to_i %>
<% if (Time.new(2022, 05, 01).to_i...Time.new(2022, 07, 31).to_i).include?(now) -%>
          img#jenkins-head-icon {
              content: url(https://www.jenkins.io/images/logos/ice-cream/ice_cream.svg);
          }

    - faviconUrl:
        url: "https://www.jenkins.io/images/logos/ice-cream/ice_cream.svg"
<% elsif (Time.new(2022, 10, 19).to_i...Time.new(2022, 10, 31).to_i).include?(now) -%>
          img#jenkins-head-icon {
              content: url(https://www.jenkins.io/images/logos/jenkinstein/jenkinstein.svg);
          }

    - faviconUrl:
        url: "https://www.jenkins.io/images/logos/jenkinstein/jenkinstein.svg"
<% elsif (Time.new(2022, 12, 01).to_i...Time.new(2022, 12, 25).to_i).include?(now) -%>
          img#jenkins-head-icon {
              content: url(https://www.jenkins.io/images/logos/santa-claus/santa-claus.svg);
          }

    - faviconUrl:
        url: "https://www.jenkins.io/images/logos/santa-claus/santa-claus.svg"
<% else -%>
          img#jenkins-head-icon {
              content: url(https://www.jenkins.io/images/logos/jenkins/jenkins.svg);
          }

    - faviconUrl:
        url: "https://www.jenkins.io/images/logos/jenkins/jenkins.svg"
<% end -%>
```



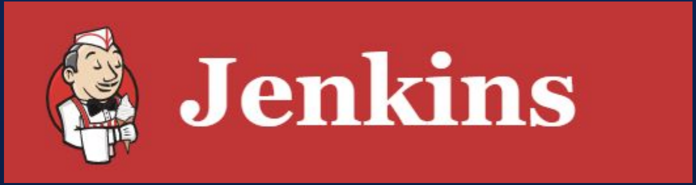
Test Kitchen



Development



QA



Production

Real World Scenario

Seamless Recovery: A Real World Scenario

- **The Incident: Volume Mounting Issue on the Host**
  - Unexpected Host Issue
  - Potential for Significant Jenkins
  - Inability to Deploy

- **The Recovery: Chef in Action**
  - Automatic Jenkins Reconstruction
  - Consistent Jobs & Configurations

Deploying Services in Minutes

- **Request for new Service Pipelines**
  - Templating
  - Hot Reloading
  - Up and running within minutes

Future Steps

- Deduplicate Code (Even More Templates)
- Move hacks to plugin fixes
- Test more process steps (Inspec)
- Support Ops Jenkins (Additional PolicyFile)

Lessons Learned

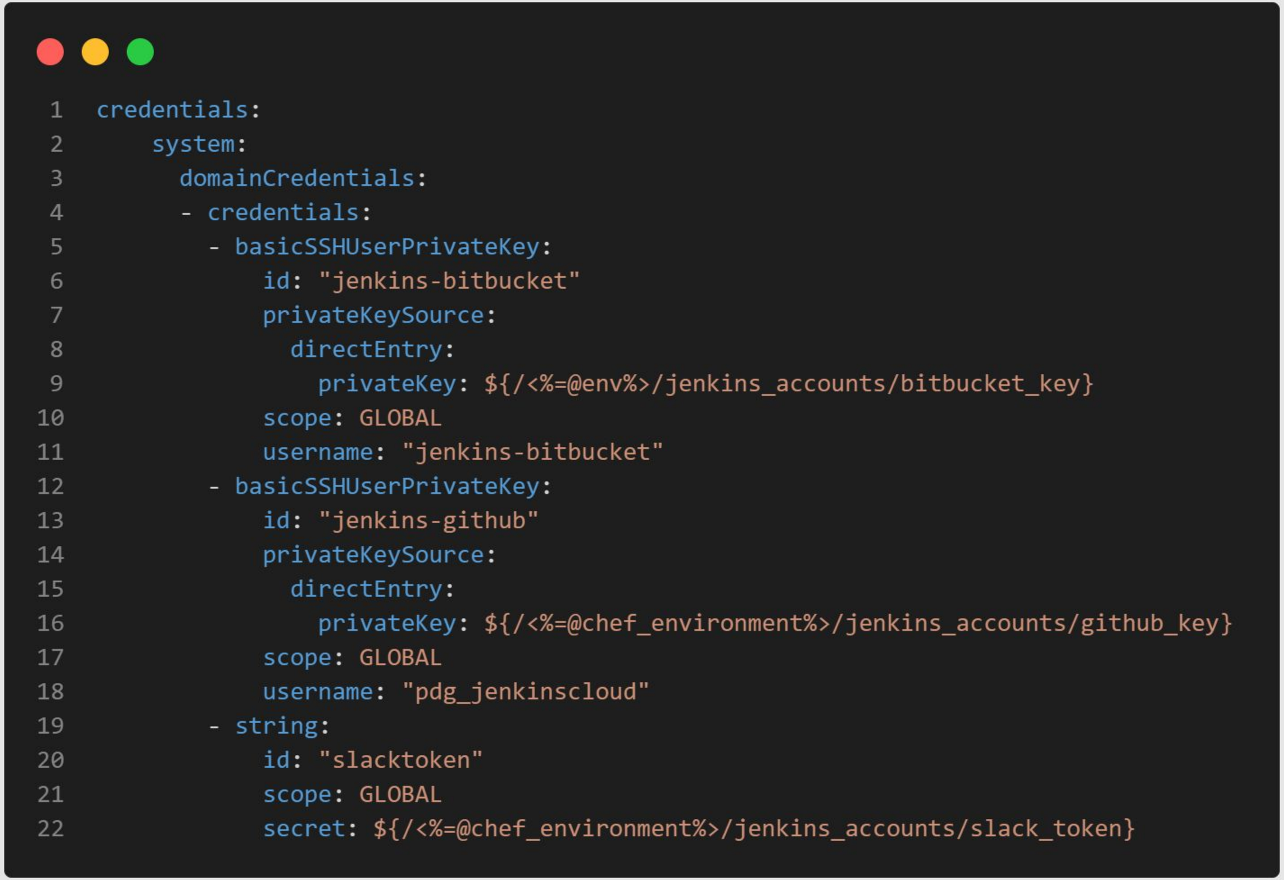
- Test Everything!
- Everything as Code
- Copy Once , Template the Third Time
- Open Source is Amazing

Q/A

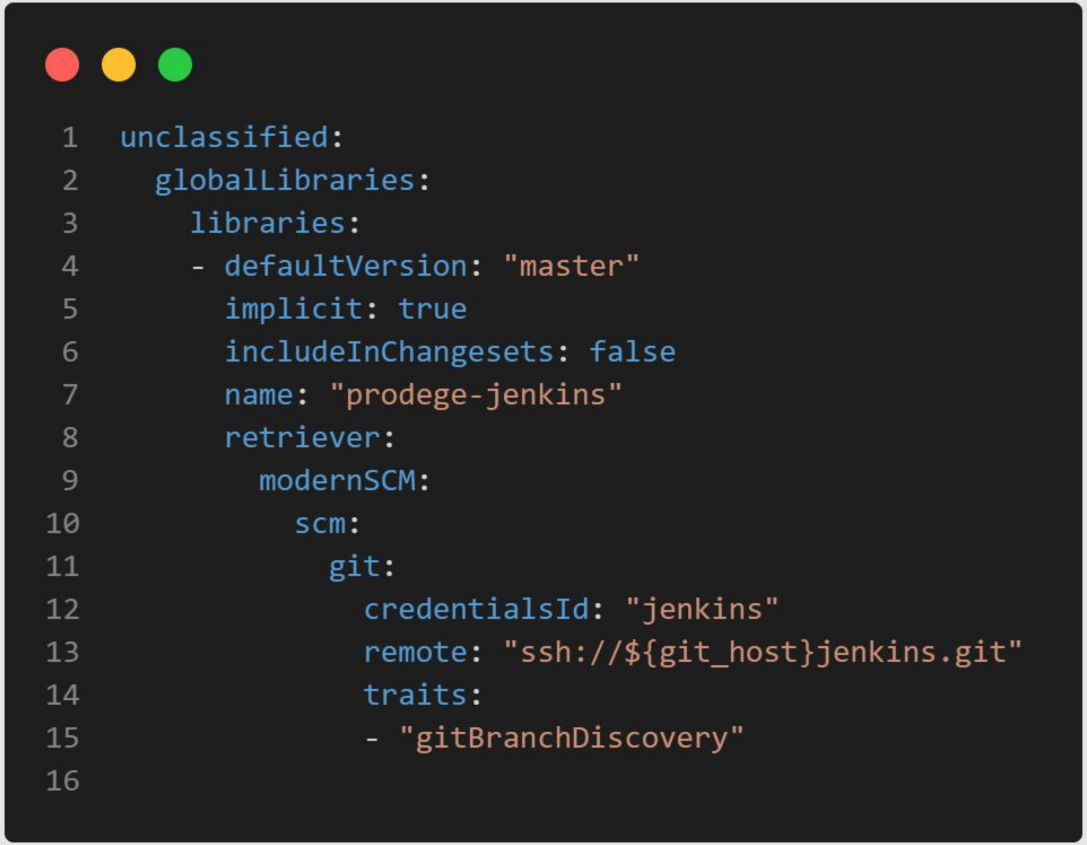
THANK YOU

```
│  kitchen.yml
│  metadata.rb
│  README.md
│
├──files
│  └──default
│     ├──dockerconfig
│     │   Dockerfile
│     │   plugins.yml
│     │
│     └──jcasc_jobs
│        │ liquibase_job.groovy
│        │ prodege-unittests_job.groovy
│        │ prodege_job.groovy
│        │ startinstance_job.groovy
│        │ staticsite_job.groovy
│        │
│        ├──codestylerules
│        │   compilejava_job.groovy
│        │   eslint_job.groovy
│        │   liquibase-linting_job.groovy
│        │   staticsitebuilder_job.groovy
│        │
│        ├──service-one
│        │   service-one-imagebuilder_job.groovy
│        │   service-one-taskdefupdater_job.groovy
│        │
│        ├──mobile
│        │   mobile-android-robot-tests_job.groovy
│        │   mobile-android_job.groovy
│        │
│        ├──prodege
│        │   prodege-imagebuilder_job.groovy
│        │   prodege-taskdefupdater_job.groovy
```

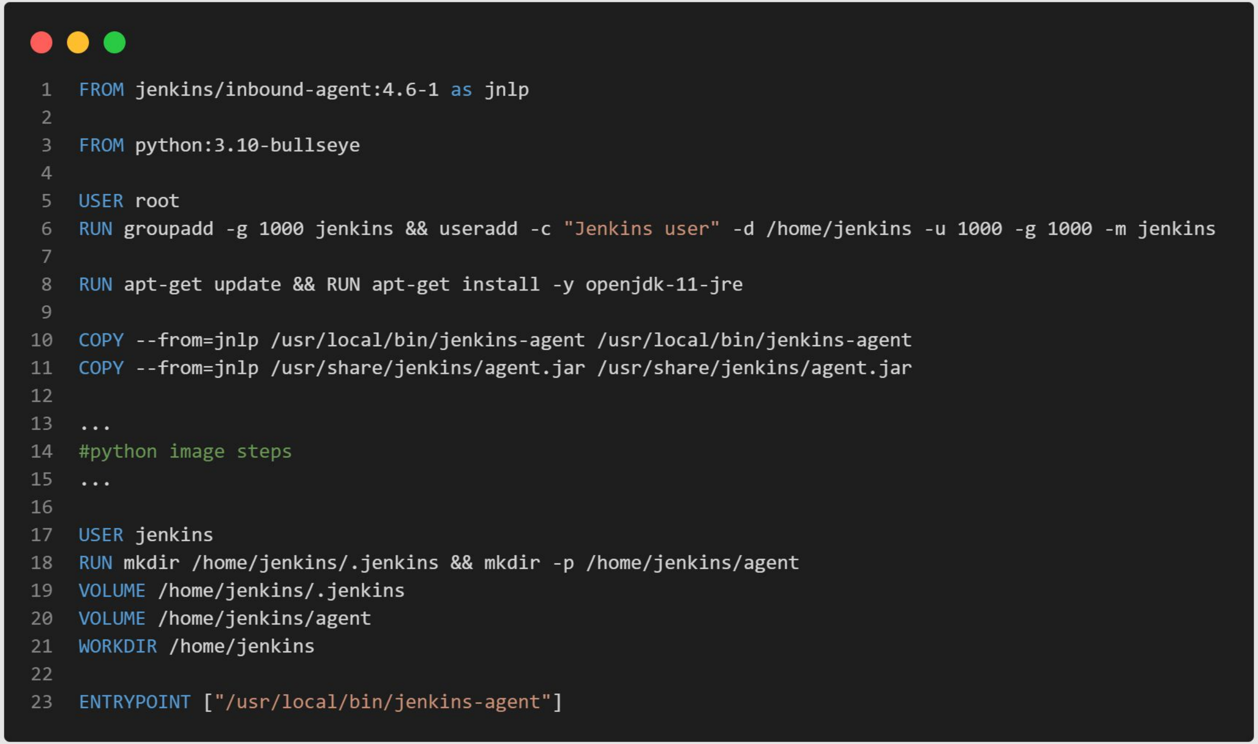
```
│        │
│        ├──service-two
│        │   service-two-imagebuilder_job.groovy
│        │   service-two-taskdefupdater_job.groovy
│        │
│        └──selenium
│            seleniumpipeline_job.groovy
│            seleniumtests_job.groovy
│
├──policyfiles
│   prdg-jenkins.rb
│
├──recipes
│   default.rb
│
├──templates
│  └──jcasc
│      docker-cloud.yaml
│      jenkins-creds.yaml
│      jenkins-jobs-codestylerules-dev.yaml
│      jenkins-jobs-codestylerules-qa.yaml
│      jenkins-jobs-service-one.yaml
│      jenkins-jobs-ec2.yaml
│      jenkins-jobs-extension.yaml
│      jenkins-jobs-lambda.yaml
│      jenkins-jobs-mobile.yaml
│      jenkins-jobs-service-two.yaml
│      jenkins-jobs-selenium.yaml
│      jenkins-jobs-tar.yaml
│      jenkins-jobs-unittests.yaml
│      jenkins-jobs.yaml
│      jenkins-libraries.yaml
│      jenkins-security.yaml
│      jenkins.yaml
│
└──test
   └──integration
      │
      └──prdg-jenkins
         │ inspec.yml
         │
         └──controls
             base_prdgjenkins.rb
```



```yaml
credentials:
    system:
      domainCredentials:
      - credentials:
        - basicSSHUserPrivateKey:
            id: "jenkins-bitbucket"
            privateKeySource:
              directEntry:
                privateKey: ${/<%=@env%>/jenkins_accounts/bitbucket_key}
            scope: GLOBAL
            username: "jenkins-bitbucket"
        - basicSSHUserPrivateKey:
            id: "jenkins-github"
            privateKeySource:
              directEntry:
                privateKey: ${/<%=@chef_environment%>/jenkins_accounts/github_key}
            scope: GLOBAL
            username: "pdg_jenkinscloud"
        - string:
            id: "slacktoken"
            scope: GLOBAL
            secret: ${/<%=@chef_environment%>/jenkins_accounts/slack_token}
```



```yaml
unclassified:
  globalLibraries:
    libraries:
    - defaultVersion: "master"
      implicit: true
      includeInChangesets: false
      name: "prodege-jenkins"
      retriever:
        modernSCM:
          scm:
            git:
              credentialsId: "jenkins"
              remote: "ssh://${git_host}jenkins.git"
              traits:
              - "gitBranchDiscovery"

```



```
FROM jenkins/inbound-agent:4.6-1 as jnlp

FROM python:3.10-bullseye

USER root
RUN groupadd -g 1000 jenkins && useradd -c "Jenkins user" -d /home/jenkins -u 1000 -g 1000 -m jenkins

RUN apt-get update && RUN apt-get install -y openjdk-11-jre

COPY --from=jnlp /usr/local/bin/jenkins-agent /usr/local/bin/jenkins-agent
COPY --from=jnlp /usr/share/jenkins/agent.jar /usr/share/jenkins/agent.jar

...
#python image steps
...

USER jenkins
RUN mkdir /home/jenkins/.jenkins && mkdir -p /home/jenkins/agent
VOLUME /home/jenkins/.jenkins
VOLUME /home/jenkins/agent
WORKDIR /home/jenkins

ENTRYPOINT ["/usr/local/bin/jenkins-agent"]
```