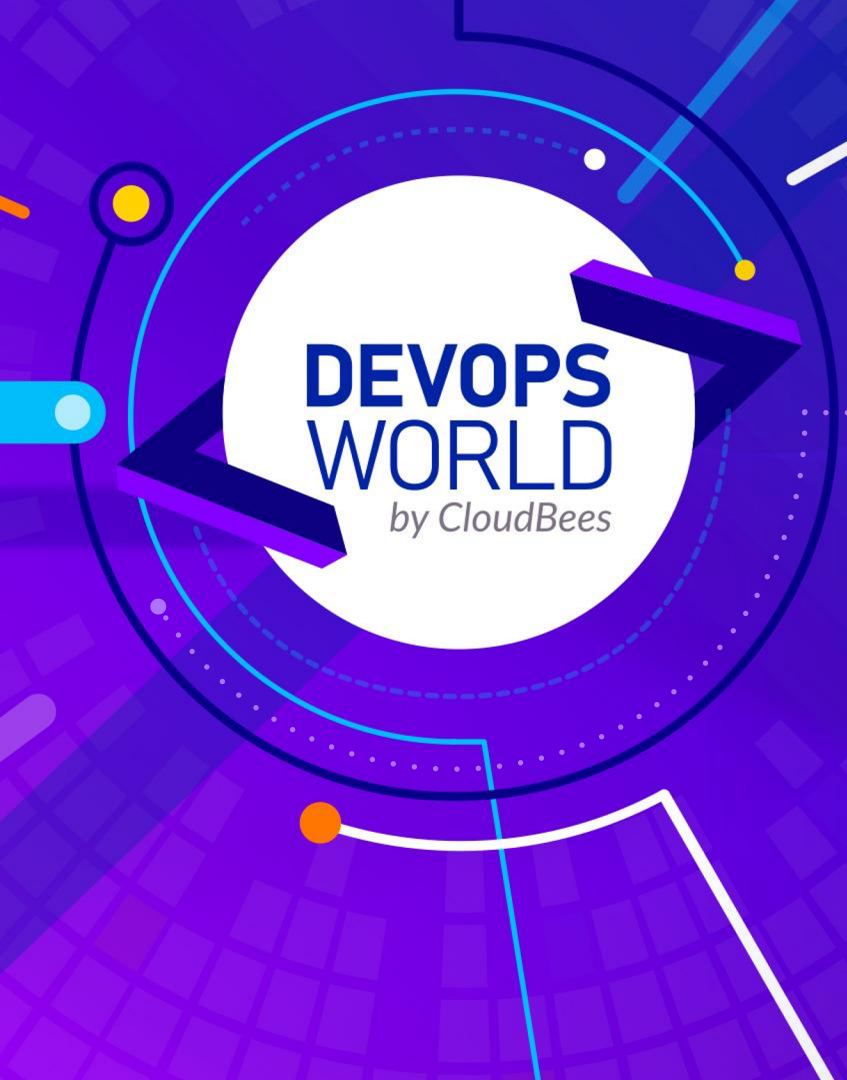


PRACTITIONER

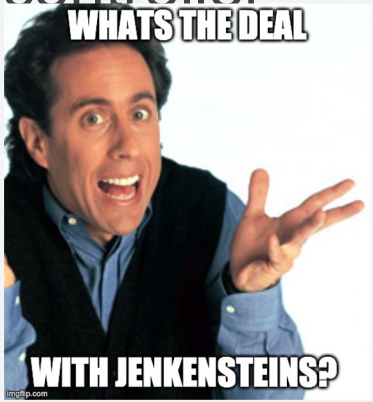
From Big and Slow to Small and Agile: Splitting Monolithic Jenkins Controllers for Increased Performance

Dylan Dewhurst

Sr. Developer Support Engineer at CloudBees

What's the Deal with Monolithic Jenkins Controllers?

- What is it? The overloaded, overcrowded Jenkins controller that many teams use for all their jobs
- Fairly common occurrence at large enterprise companies
- Easier for Admins to onboard onto existing controller
- This creates problems down the road



PRACTITIONER

Monolith Vs. Horizontal Scaling

Why Monolithic is A Bad Practice

- The main issue is poor performance
  - Slow startup, slow UI, long running builds
- This often leads to vertical scaling:
  - More CPU
  - More memory
  - More disk space

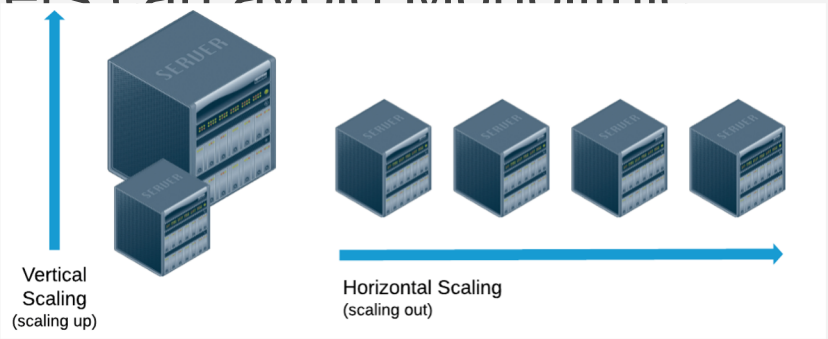
- Outages affect many users and down time is usually increased

Why Monolithic is A Bad Practice

- Jobs affecting performance are harder to find
  - Too many builds running simultaneously
  - Many nested folders of jobs
- Too many plugins. Many are out of date.
  - Sometimes plugins haven't been updated in years
  - There may be plugins not even used
- Managing permissions becomes harder

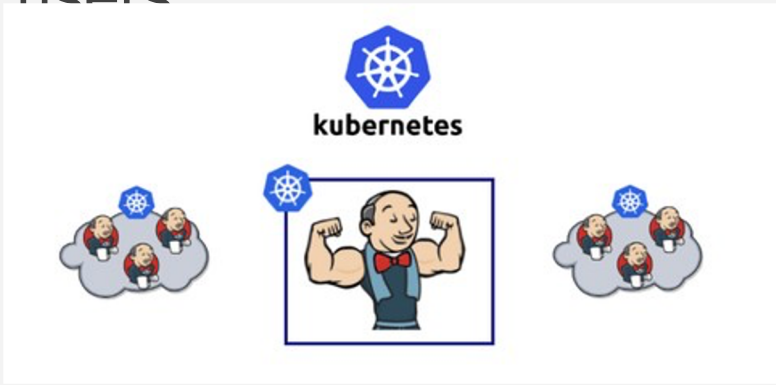
Horizontal Scaling is the Answer!

- Many smaller Jenkins controllers can avoid Monolithic problems
  - Faster startup
  - Faster build times
  - Faster UI
- Smaller number of users, typically one team
  - Could self manage
- Less plugins to maintain



Horizontal Scaling is the Answer!

- An outage is shorter and affects less users
- Easier to find bad plugin or job



- Great for Kubernetes!
  - For CloudBees CI, easily managed by Operations Center
- This all leads to happier users and Admins!

Why Kubernetes?

- Containers work great with single applications
  - Fast to start
  - Easy to move, change config and resources
  - Many fit on one server, easily scalable
- Cloud ready with EKS, GKE, AKS
- Ephemeral Agents that can scale on demand
- Kubernetes is the future of DevOps!

PRACTITIONER

Guidelines for Scaling

DEVOPSWORLD
by CloudBees

Scaling Guidelines - Sizing

- Keep user count low per controller
  - Preferably one team or business unit
- Limit heap size to 16GB maximum
- As best practice, no more than 5000 jobs per controller
- No executors on controller
  - Use Agents for all builds

Scaling Guidelines - Configuration

- Use Configuration as Code plugin
  - For CloudBees CI, Configuration as Code for Controllers
- Limit use of Groovy in Pipeline jobs and Script Console
  - Could heavily impact performance
- Set build discarder policies
- Use webhooks over polling

PRACTITIONER

Preparing to Horizontally Scale

DEVOPSWORLD by CloudBees

Preparing Existing Controller

- Data cleanup
  - Configure build discard policy. Remove unused jobs.
  - Remove unused plugins. If a CloudBees CI customer, can use CloudBees Plugin Usage Analyzer plugin.
- Update in use plugins
- Review Jenkins health using Health Advisor by CloudBees plugin
  - Prevent duplicating issues when scaling

Creating New Jenkins Controller

- Create on a server or container with less resources
  - May require a bit of trial and error to get right
- Use same Jenkins version and same/similar arguments, variables
  - If using containers, most configuration can be reused
- Start the new controller for the first time for initial setup

PRACTITIONER

Data Migration and Configuration

DEVOPSWORLD by CloudBees

Backup and Export

- Put Source Jenkins in Quiet Down mode: `$JENKINS_URL/quietDown`
- Export system and folder level credentials
  - Scripts provided in CloudBees "Splitting a controller" documentation
- Backup `$JENKINS_HOME` directory
  - If a CloudBees CI customer, can use the CloudBees Backup plugin
  - Make sure to only backup the jobs in `$JENKINS_HOME/jobs` that are needed for the new controller.

Backup - Excludes

- Exclude the following files from your backup as they are unique to each Jenkins controller:

```
secret.key
license.xml
identity.key.enc
jgroups/
operations-center-cloud*
operations-center-client*
com.cloudbees.opscenter.client.plugin.OperationsCenterRootAction.xml
nodes/
secrets/master.key
```

Restoring at Destination

- Copy the backup data into the `$JENKINS_HOME` of the new destination controller
  - If a CloudBees CI customer, can use a restore job
- Import the system and folder level credentials from source controller
  - Use encoded output from export script
  - Scripts provided in CloudBees "Splitting a controller" documentation

Additional Configuration

- After restore, a number of items need to be reconfigured:
  - Agents
  - Authentication (possibly)
  - Webhooks
  - Jenkins URL
  - Third Party Integrations
  - Another plugin review

Resources

- More detailed steps covered in: CloudBees Splitting a Controller
- Configuration as Code plugin
- Health Advisor by CloudBees
- Quiet Down mode
- Kubernetes Plugin
- Pipeline Best Practices
- Best Strategy for Disk Space Management
- SCM Best Practices

Thank you!

Q&A

Dylan Dewhurst

Sr. Developer Support Engineer