

# **HOW NETFLIX AUTOSCALES CI**

Rahul Somasunderam

# **WHAT DOES CI LOOK LIKE AT NETFLIX**

# JENKINS @ NETFLIX

- 35 Jenkins controllers
- ~45k job definitions
- ~600k builds per week
- 650-1500 agents
- 1-100 executors per agent

# THE SPINNAKER VIEW

- 1 Application
- 35 stacks (Controller Clusters)
- 180 Agent Clusters
- 1+ ASG per cluster
- All workloads on AWS

# **CLUSTERS AND ASGS**

# CLUSTERS AND ASGS

- AWS has Auto Scaling Groups

# CLUSTERS AND ASGS

- AWS has Auto Scaling Groups
- Spinnaker calls them Server Groups

# CLUSTERS AND ASGS

- AWS has Auto Scaling Groups
- Spinnaker calls them Server Groups
- <Application>-<Stack>-<Detail>-v<Version>

# CLUSTERS AND ASGS

- AWS has Auto Scaling Groups
- Spinnaker calls them Server Groups
- <Application>-<Stack>-<Detail>-v<Version>
- jenkins-unstable-agent-highlander-v123

# **HOW TO PLAN FOR CI INFRASTRUCTURE**

# INFINITE RESOURCES

- Provision capacity based on known maximum load
- Multiply by a safety factor for good measure
- Monitor and change the capacity as load increases

# **INFINITE PATIENCE**

- Plan capacity based on median load
- Builds will sit in queue for long times

# INSTANT RESOURCES

- You will get resources as soon as you request for them
- Works well with Containerizable builds
- Not all builds can be containerized
- Does not scale well with large numbers of short-lived builds

# AUTOSCALING

- Set up minimum and maximum capacity
- Scale based on some metric

# **WHAT METRIC TO USE**

# **SYSTEM METRICS**

# SYSTEM METRICS

CPU/Memory/Disk IO/Network throughput

- Natively supported by cloud providers and most metrics solutions

# SYSTEM METRICS

CPU/Memory/Disk IO/Network throughput

- Natively supported by cloud providers and most metrics solutions

Scaling Policies are supported by cloud providers

# **SYSTEM METRICS**

Not very useful for CI

# **QUEUE DEPTH**

# **QUEUE DEPTH**

Queue Depth seems adequately proportional.

# **QUEUE DEPTH**

Queue Depth seems adequately proportional.  
However, it is a trailing metric.

# **AGENT UTILIZATION**

# **AGENT UTILIZATION**

For each agent, find [idle, busy, offline] executors.

# AGENT UTILIZATION

For each agent, find [idle, busy, offline] executors.

Sum these up by ASG.

# AGENT UTILIZATION

For each agent, find [idle, busy, offline] executors.

Sum these up by ASG.

Compute utilization as  $\frac{busy + offline}{busy + offline + e}$

# **MEASURING AGENT UTILIZATION**

# AN AGENT'S ASG

When launching agents, use labels to specify the placement of the agent.

 Agent **nflux-agent-unstable-i-0522989245ff3659d** (Connect: `ssh -t i-0522989245ff3659d`)

Mark this node temporarily offline

Agent is connected.

### Labels

```
asg:jenkins-unstable-bionic-v189 aws:test:us-east-1:jenkins-unstable-bionic-v189 bionic buildgroup:bionic carson.version:0.767.0 carson:true cloud:aws cluster:jenkins-unstable-bionic detail:bionic ec2.availZone:us-east-1e ec2.instanceType:m5d.xlarge ec2.region:us-east-1 env:test executors:4 iamRole:jenkinsInstanceProfile java.jvm:zulu8 java.runtime:1.8.0_292-b10 nf.account:test nf.app:jenkins nflux.agent.build:569 os.arch:amd64 os.codename:bionic os.distribution:ubuntu os.name:linux os.release:18.04 stack:unstable us-east-1
```

# CAPTURING METRICS

We wrote a custom plugin that plays well with Atlas.  
You could write one for whatever your metrics  
capturing service is.

# **AUTOSCALING**

# HOW TO AUTOSCALE

AWS offers 2 ways to scale

- Target Tracking
- Step Scaling

# WHEN TO SCALE UP

Edit scaling policy X

**Conditions**

Whenever  of  [Search all metrics](#) ?

is   None

for at least  consecutive period(s) of

# HOW TO SCALE UP

## Actions

Add	▼	20	percent of group ▼	when <b>jenkins.executorsUtilization</b> is between 0.65 and	0.8
Add	40	percent of group	when <b>jenkins.executorsUtilization</b> is greater than or equal to 0.8		
 Add step					

 Documentation

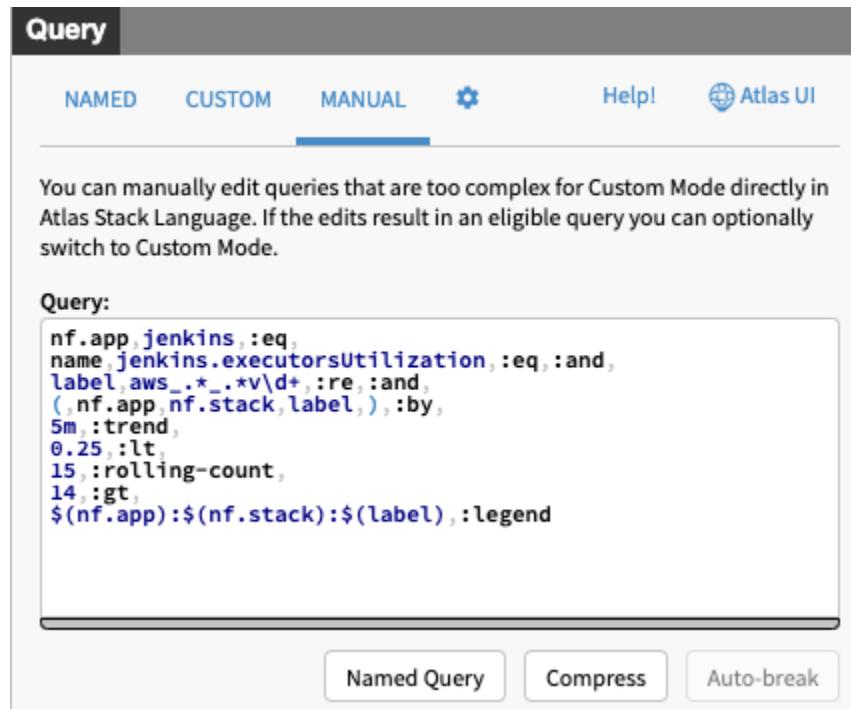
## Additional Settings

**Policy Name** jenkins-buildstest-bionic\_classic-v030-NFLX/EPIC-jenkins.executorsUtilization-GreaterThanThreshold-0.65-1-60-1620084562030

**Adjustment Step** Add instances in increments of at least  instance(s)

**Warmup** Instances need  seconds to warm up after each step

# WHEN TO SCALE DOWN



The screenshot shows the MongoDB Atlas Query interface. The top navigation bar has tabs for 'NAMED', 'CUSTOM', 'MANUAL' (which is underlined), 'Help!', and 'Atlas UI'. Below the tabs, a message reads: 'You can manually edit queries that are too complex for Custom Mode directly in Atlas Stack Language. If the edits result in an eligible query you can optionally switch to Custom Mode.' The main area is labeled 'Query:' and contains the following query text:

```
nf.app,jenkins :eq,  
name,jenkins.executorsUtilization :eq,:and,  
label,aws_*.*v\d*:re,:and,  
(,nf.app,nf.stack,label):by  
5m:trend  
0.25:lt  
15:rolling-count  
14:gt  
$(nf.app):$(nf.stack):$(label):legend
```

At the bottom of the query editor, there are three buttons: 'Named Query', 'Compress', and 'Auto-break'.

# HOW TO SCALE DOWN

# **RECAP**

# **WHAT WE LEARNT**

# **WHAT WE LEARNT**

- This improved support experience

# **WHAT WE LEARNT**

- This improved support experience
- This improved the experience for spiky workloads

# **THANK YOU!**

[jobs.netflix.com](http://jobs.netflix.com)