Scaling Jenkins at Scale

Oleg Zheltyannikov

Enterprise Jenkins Tech Lead
@Capital One

About Me

- Manage Jenkins fleet at Capital One
- Doing DevOpsy things for last 12+ years
- Make non cloud applications work in AWS for 6+ years

Agenda

- Challenges of managing Jenkins at Capital One scale
- Containerization as silver bullet
- Running Jenkins agents as AWS ECS tasks

Jenkins at Capital One

8k
Developers

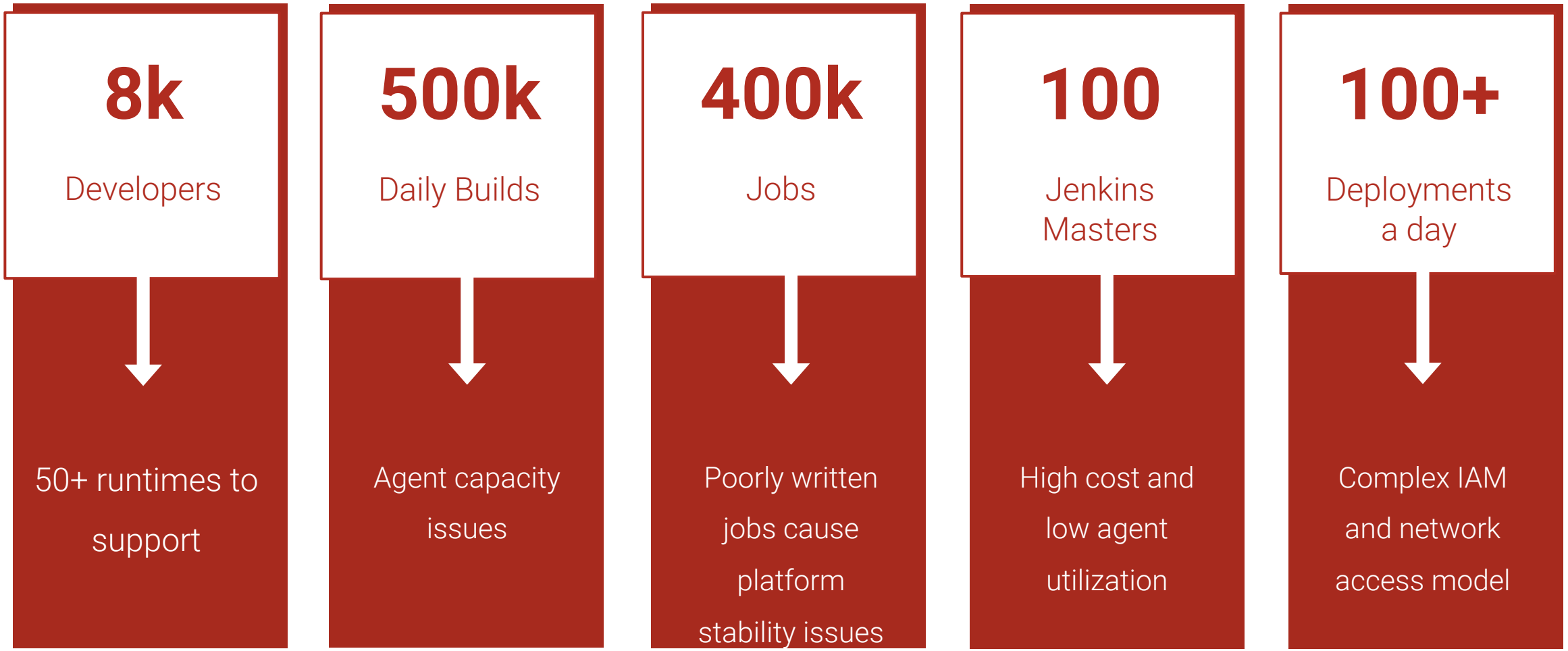
500k
Daily builds

400k
Jobs

100
Jenkins Masters

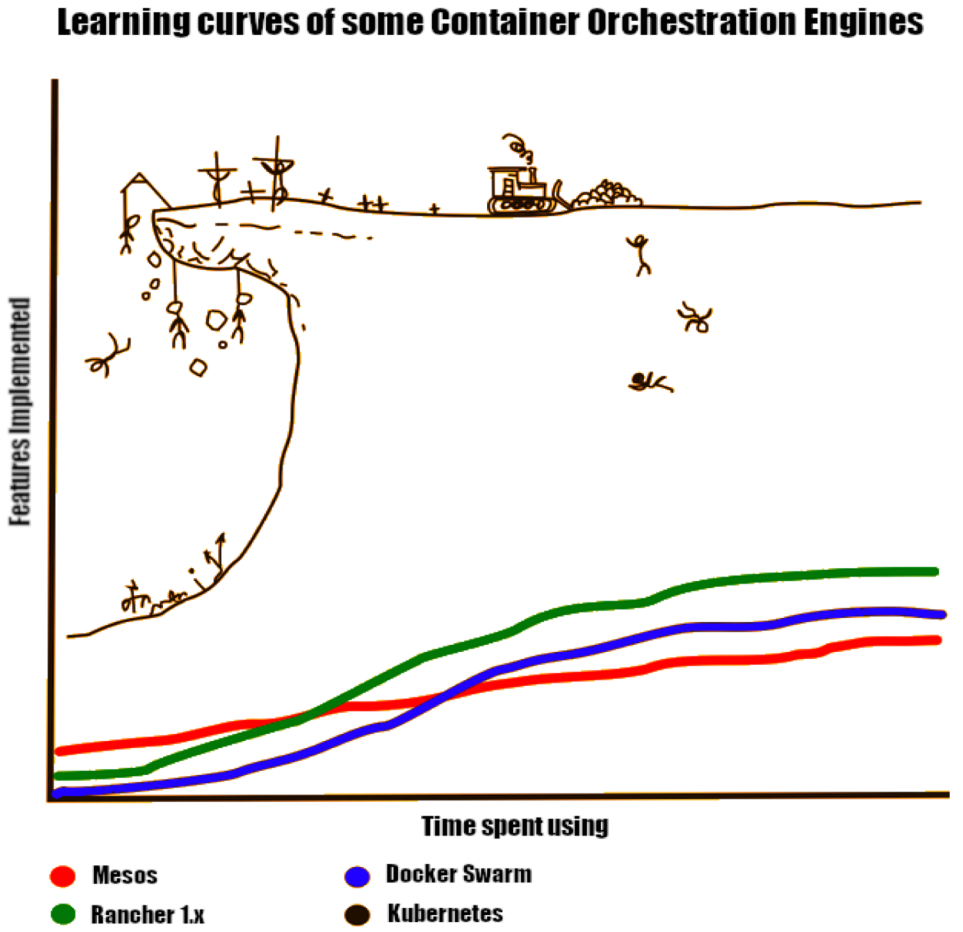
100+
Deployments a day

Challenges of Scale

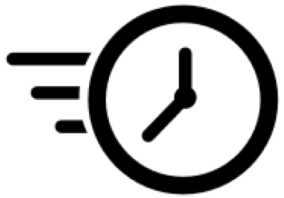


Containerization as a Silver Bullet

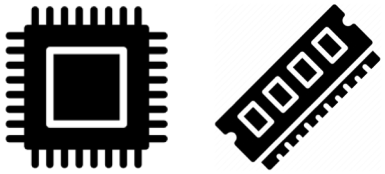
- Why containerize?
- What to choose for container orchestration?



Why containerize?

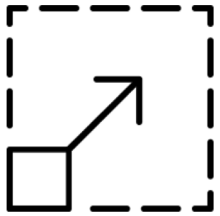


Easier runtime management

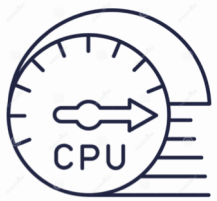


CPU/memory limits per build

Repeatable builds

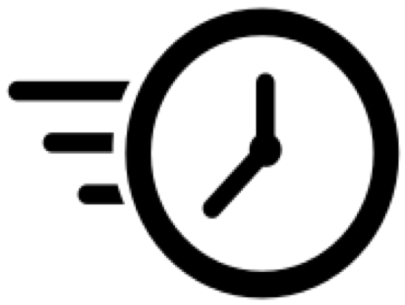


Quicker agent provisioning



Higher resource utilization

Why containerize?



Easier runtime management

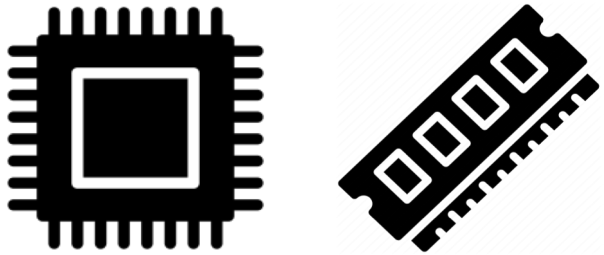
Simplified agent configuration

```
FROM <put your jnlp image here>

# install base OS packages
ENV YUM_PACKAGES="python python-devel python-setuptools"
RUN yum --assumeyes install $YUM_PACKAGES && yum clean all
```

Simplified rollout/rollback

```
images:
  rhel7: private.repository/jenkins/jenkins-agent:v1.1.0
```

Why containerize?



CPU/memory limits per build

CPU and memory limits

--cpu-shares N:
Configures relative CPU weight to container. 1 CPU = 1024 shares.
Unallocated CPU shares are shared across all other containers on a host

--memory-reservation N:
Soft memory limit. Used mostly for reservation and planning purpose by container engine.

--memory N:
Hard memory limit. Max amount of memory container can use unless swap is configured

--memory-swap N:
Container swap size. If container uses more than a memory hard limit it will start using swap

Why containerize?

Repeatable builds

Stateless/one shot agents

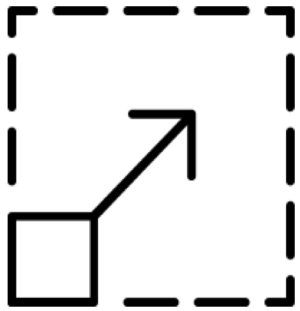
Global configurations can significantly impact behavior of your build. This is just a tiny subset of folder which are created automatically while installing build dependencies.

```
~/.gem  ~/.aws  ~/.docker  ~/.node_modules/
```

Building software with existing cache and without cache can yield significantly different results.

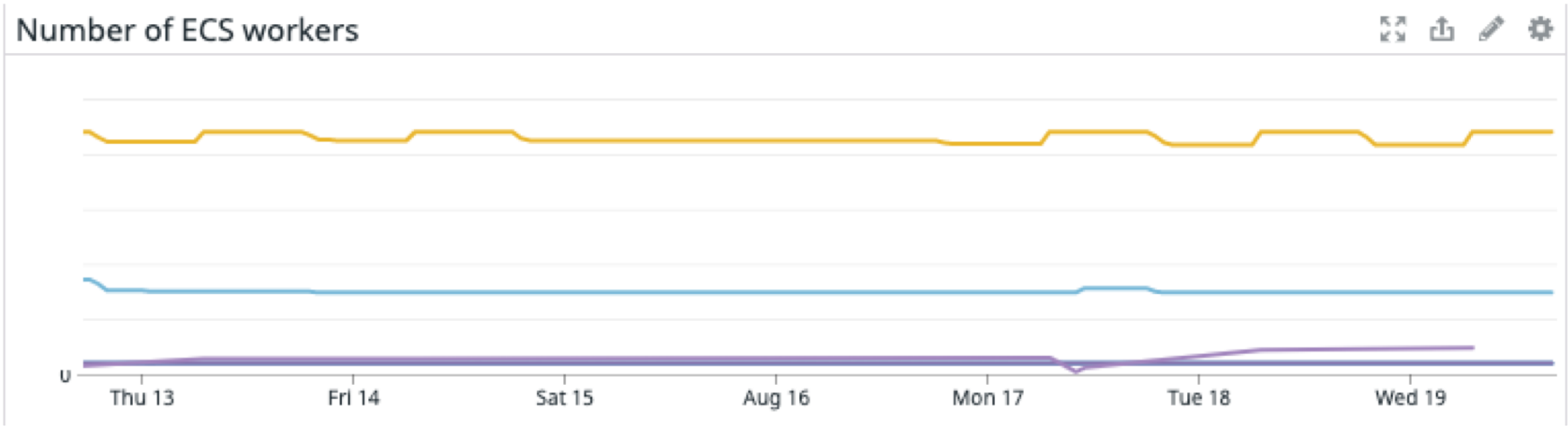
With containers no state is preserved between runs so your build goes through the same steps regardless of results of previous builds.

Why containerize?

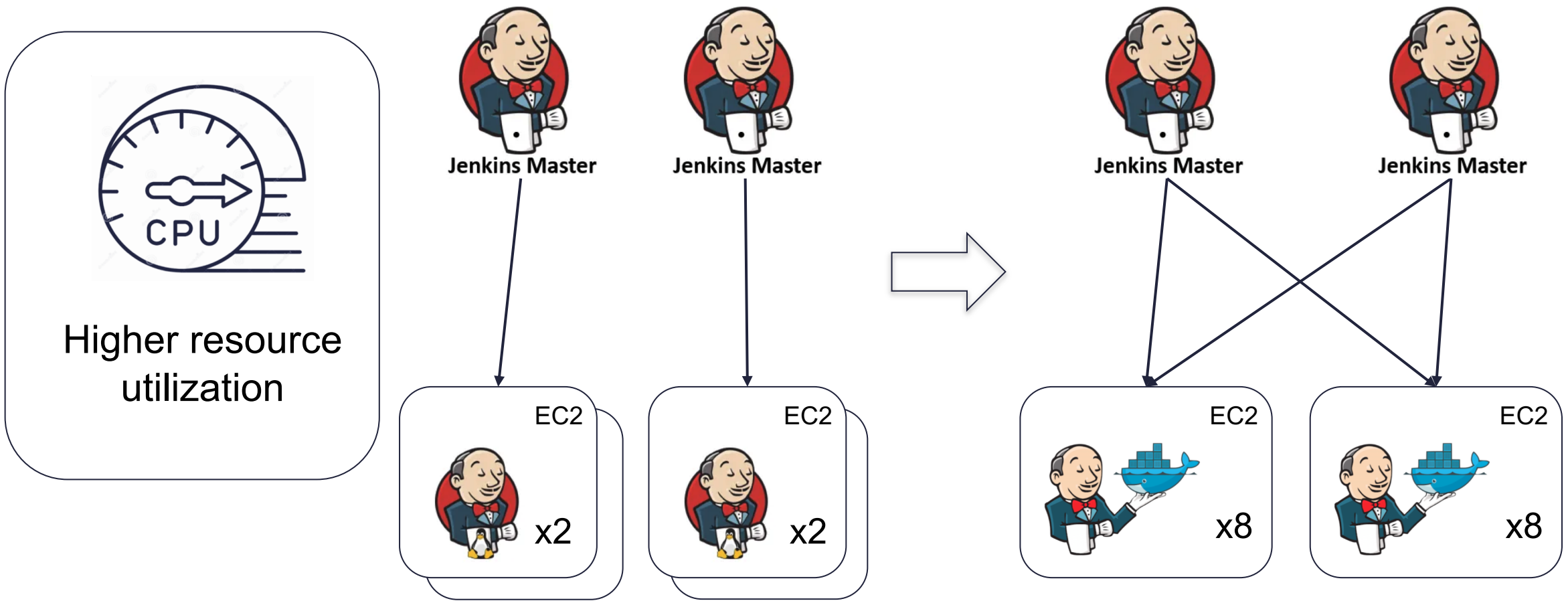


Quicker agent provisioning

Dynamic scaling during off hours and before business hours



Why containerize?



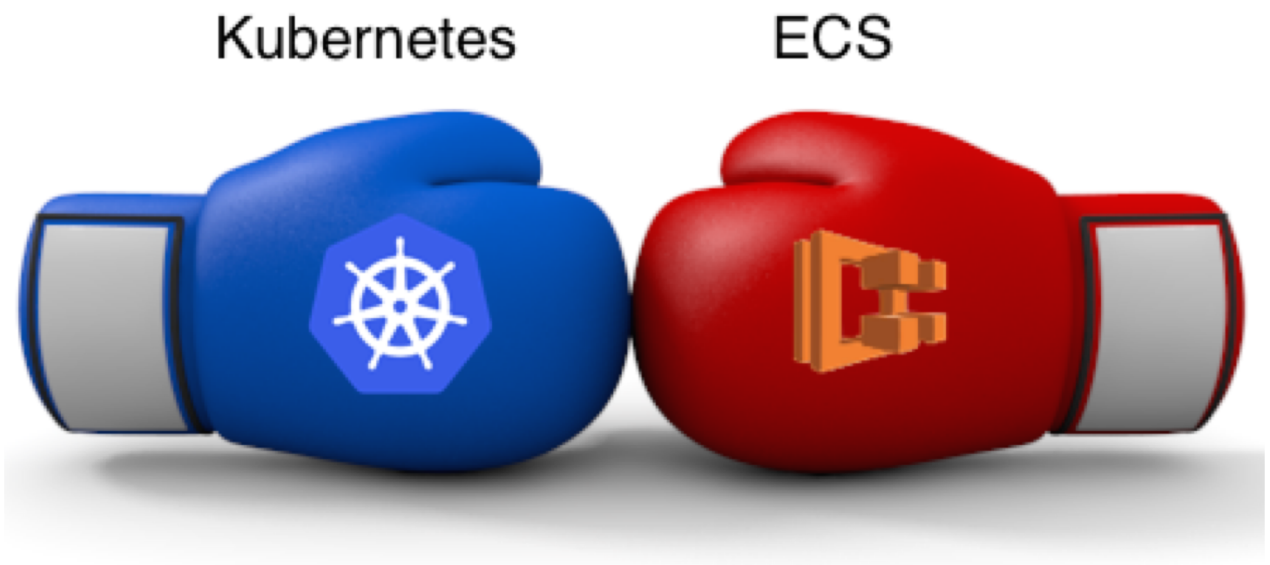
What to choose for container orchestration?

Pros:

- Mature solution with a lot of features
- Everyone uses it

Cons:

- Complex to support
- Doesn’t integrate natively with AWS



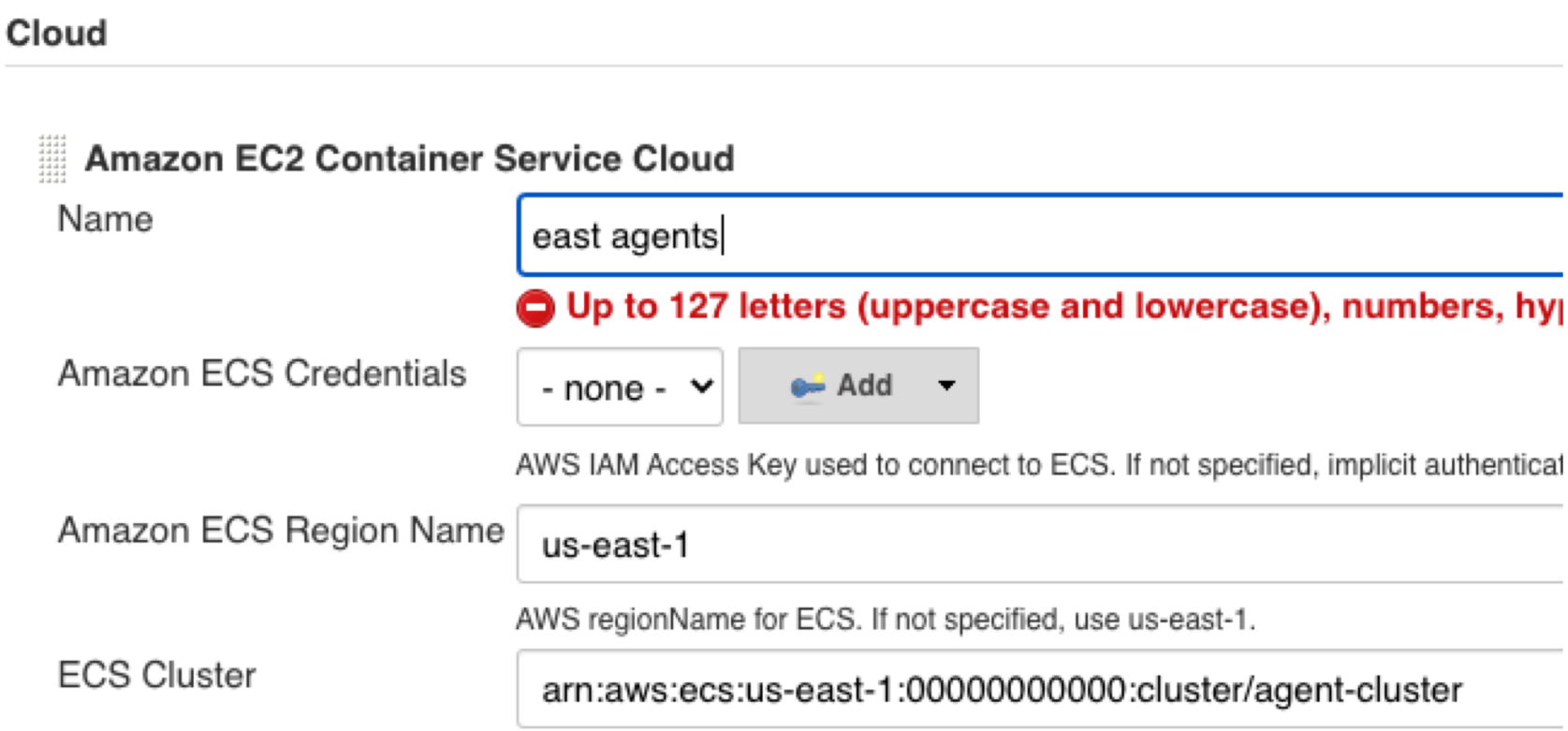
Pros:

- Easy to manage
- Integrates well with AWS network and IAM constraints

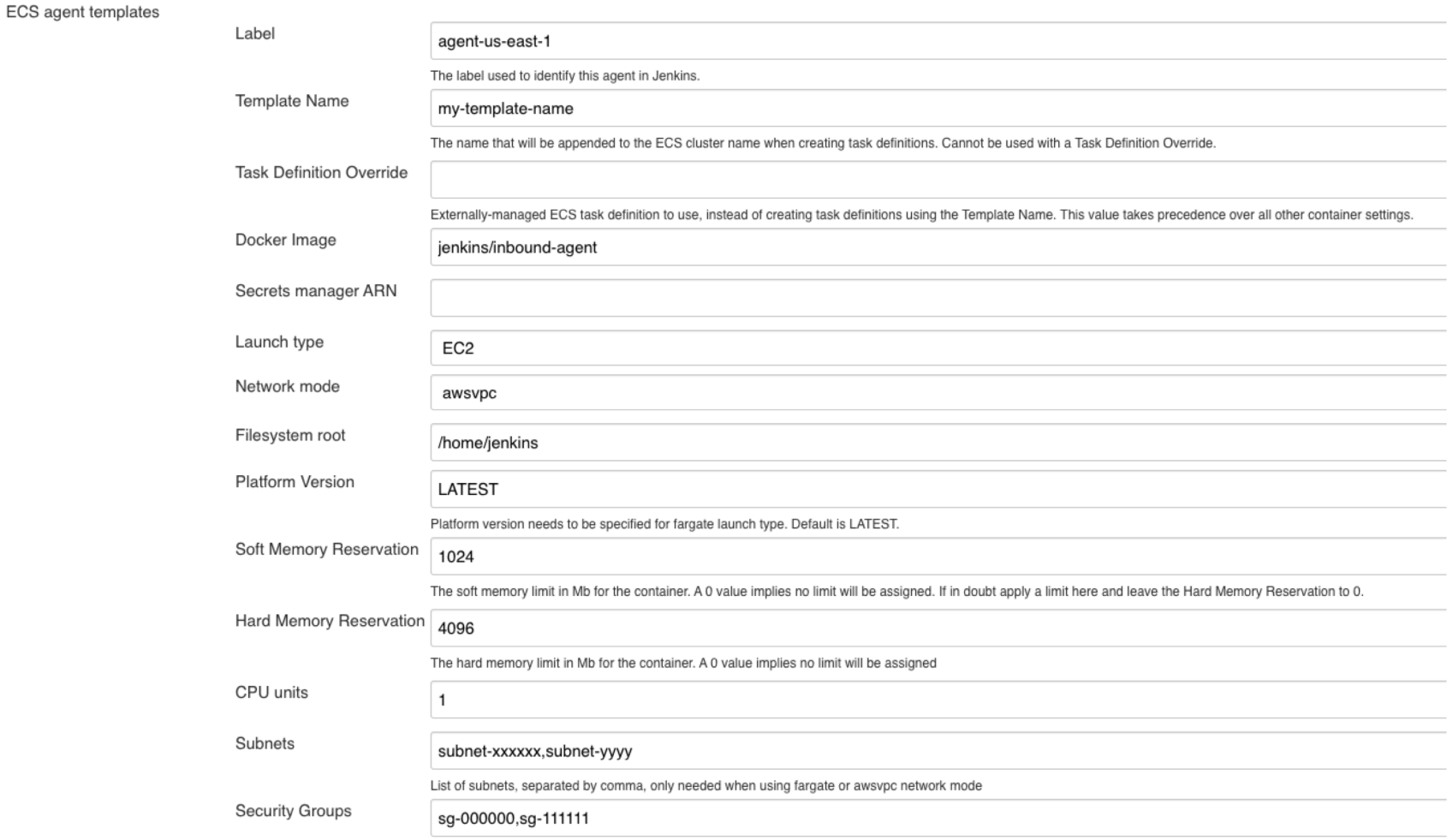
Cons:

- Very simple scheduler

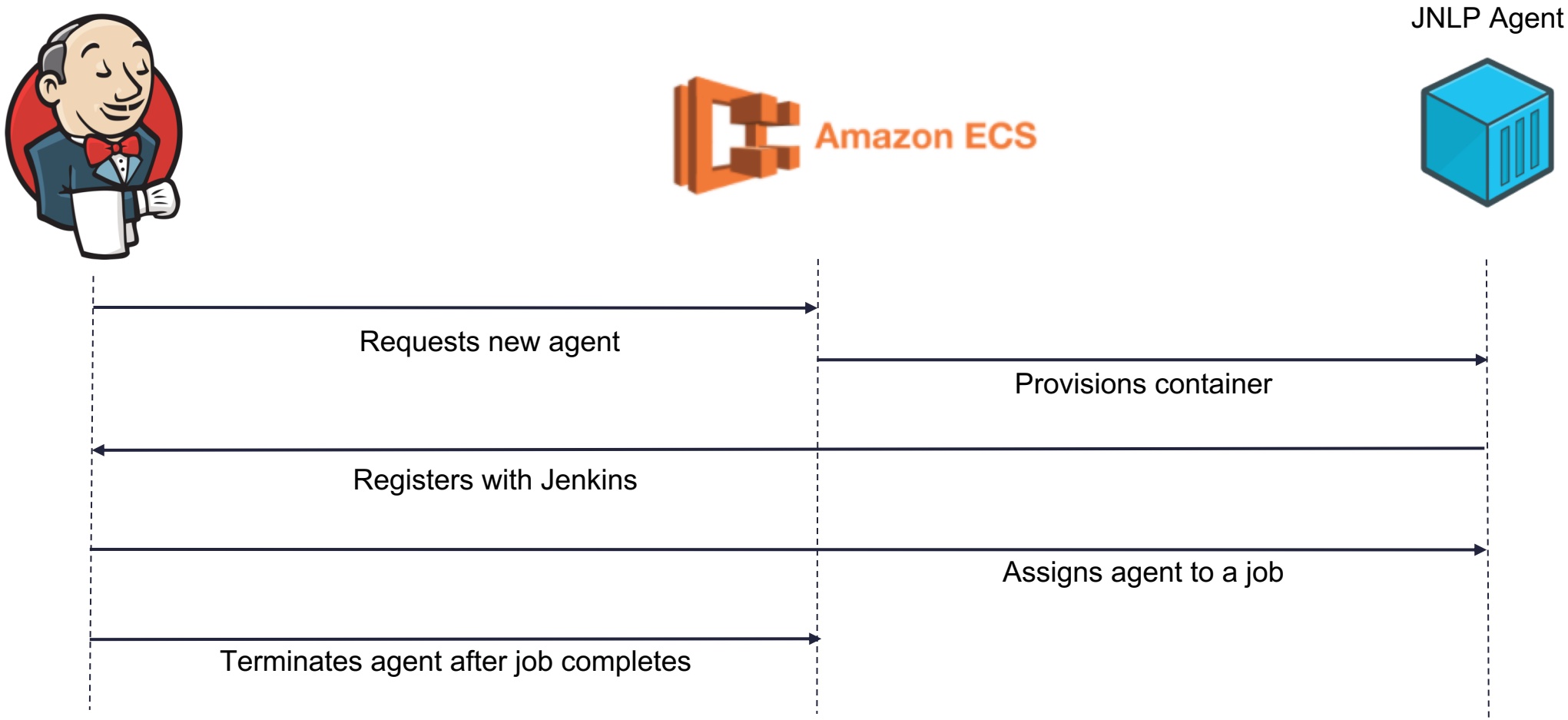
ECS - Platform for Agents



ECS - Platform for Agents



ECS - Platform for Agents



ECS - Platform for Agents

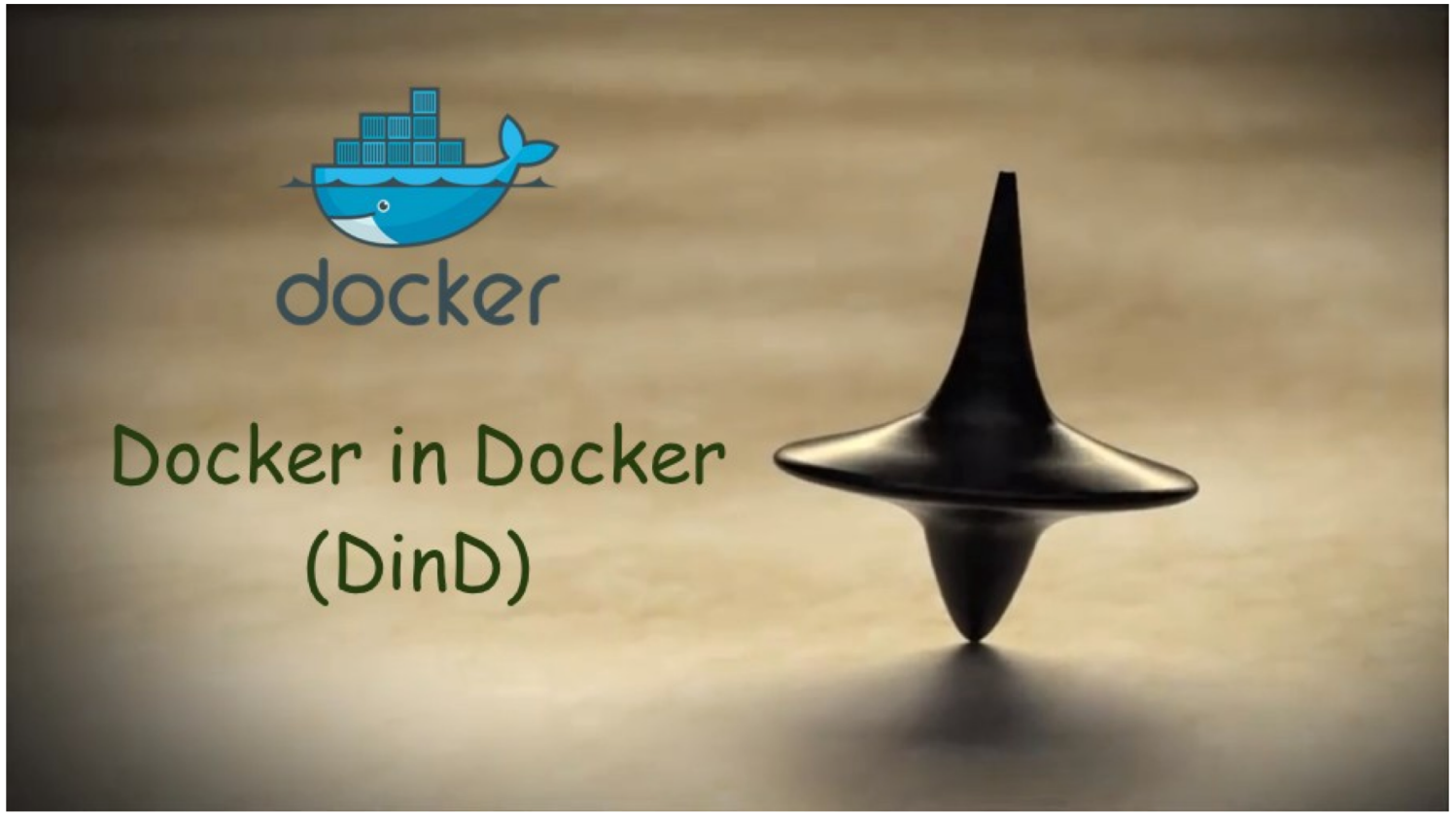
Agent per pipeline

```
pipeline {
    agent { label 'aws-agents' }

    stage('A') { ... }
    stage('B') { ... }
}
```

Agent per stage

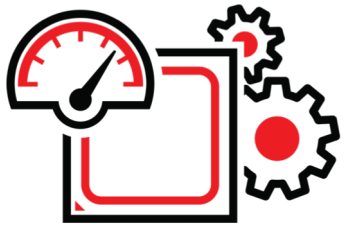
```
pipeline {
    agent none

    stage('A') {
        agent { label 'aws-agents' }
    }

    stage('B') {
        agent { label 'aws-agents' }
    }
}
```

ECS - Platform for Agents



Increased Containerization = High Rewards

Happy Jenkins team



900% increase in resource utilization

3x decrease in cost

Horizontal scaling of agents

Happy Developers

AWS ECS specific recommendations

- Analyze your workflow and pick appropriate instance type
  We tried different c5 families and picked c5d.4xlarge. It works best for us from CPU and IOPS perspective

- Check what kind of network mode do you need
  Awsvpc mode provides full integration with AWS IAM and security groups but has performance impact on agent provisioning. Penalty for using awsvpc mode could be 30-60 seconds depending how busy your AWS account

- Optimize your container size
  If you container is too big you may want to implement additional logic to pull container on the server before Jenkins starts using. Otherwise your job start time can be impacted

AWS ECS specific recommendations

- Jobs may require modifications if they use EC2 metadata endpoint
  EC2 metadata are not available in ECS environment. You may need to rewrite your jobs to use ECS metadata endpoint

Additional resources

- Another take on ECS Jenkins agents from DevopsDays Austin 2019: https://www.youtube.com/watch?v=9BpS50YVqdw
- ECS Plugin: https://github.com/jenkinsci/amazon-ecs-plugin
- Kaniko from Google: https://github.com/GoogleContainerTools/kanik
- How to do docker in docker: https://medium.com/faun/docker-in-docker-the-real-one-e54133639c55
- Docker in docker as a sidecar on k8s: https://applatix.com/case-docker-docker-kubernetes-part-2/

Q&A

Thanks for watching.

If you have any question about this talk please reach out it to me via https://www.linkedin.com/in/ozheltyannikov/