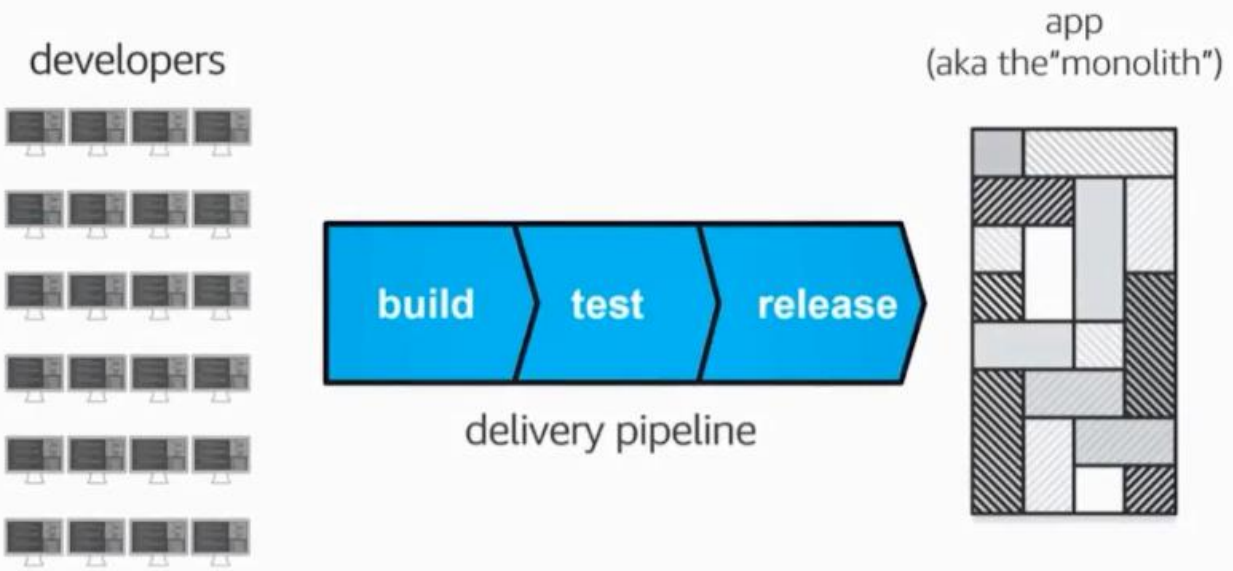
Containers allow you to easily package an application's code, configurations, and dependencies into easy to use building blocks that deliver environmental consistency, operational efficiency, developer productivity, and version control. But how can developers leverage containers to drive innovation for their applications, their team, and organization?

In this session, Asif Khan Technical Business Manager for AWS will discuss how containers are becoming a new cloud native compute primitive, and how your organization can use containers as a building block to accelerate innovation.

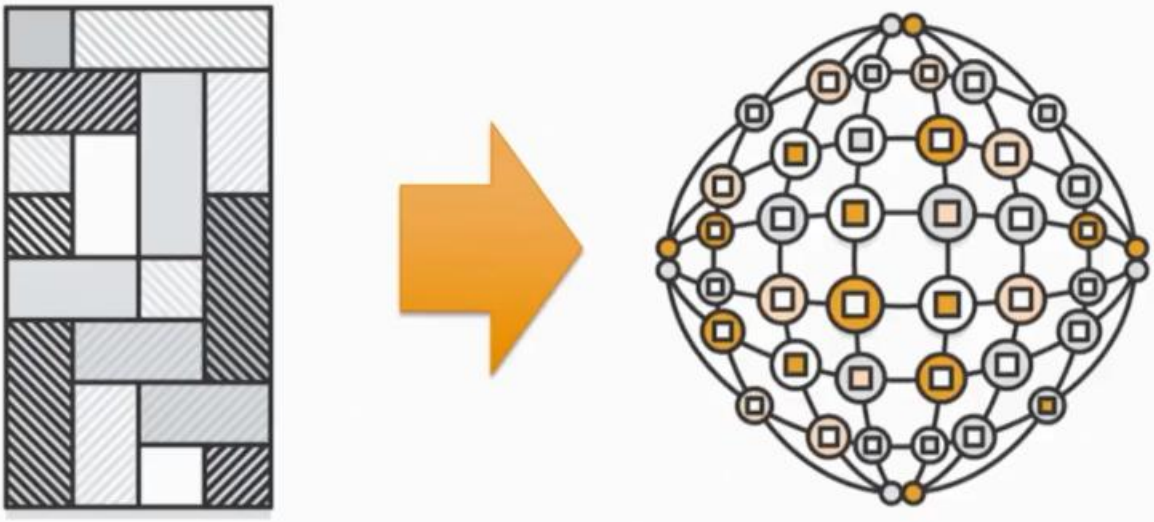
WeWork's Christopher Tava, Joshua Davis, and OpsLine's Radek Wierzbicki will show how they adopted containers as discipline in code development, and how they refactored their production architecture into containers running on Amazon ECS in under 8 months.

Traditional Software Development and Deployment



Challenges with the traditional software development lifecycle

Infrastructure	Application	Business
Difficult to scale	Architecture is hard to maintain and evolve	Lack of agility
Long Build/Test/Release Cycles (who broke the build?)	New releases take months	Lack of innovation
Operations is a nightmare (module X is failing, who's the owner?)	Long time to add new features	Frustrated customers



"service-oriented architecture composed of loosely coupled elements that have bounded contexts"

- Adrian Cockcroft (VP, Cloud Architecture Strategy at AWS)

service-oriented architecture

Services communicate with each other over the network

composed of

loosely coupled elements

that have

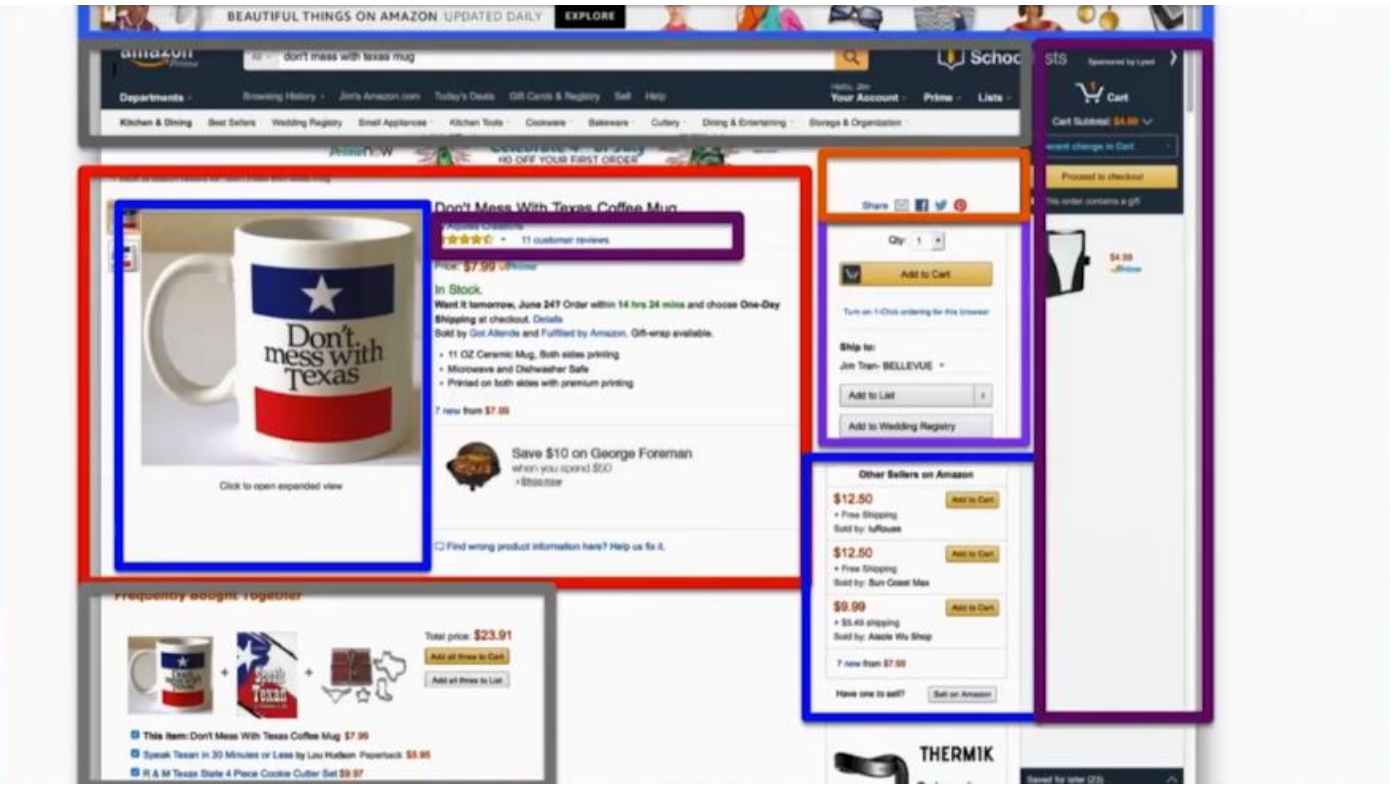
You can update the services independently; updating one service doesn't require changing any other services.

elements

that have

bounded contexts

Self-contained; you can update the code without knowing anything about the internals of other microservices



Portability

Same immutable images. Run anywhere.

Flexibility

Create modular environment. Decompose Apps.

Speed

Speeds up build and release cycles.

Efficiency

Optimize resource allocation.

Agility.

Thousands of teams

× Microservice architecture

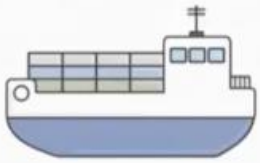
× Continuous delivery

× Multiple environments

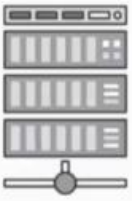
= 50 million deployments a year

5708 per hour – one every 0.63 seconds!

BUILD

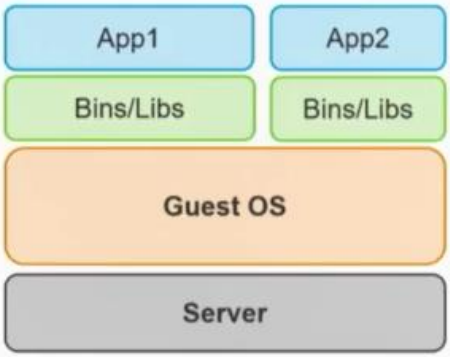


SHIP

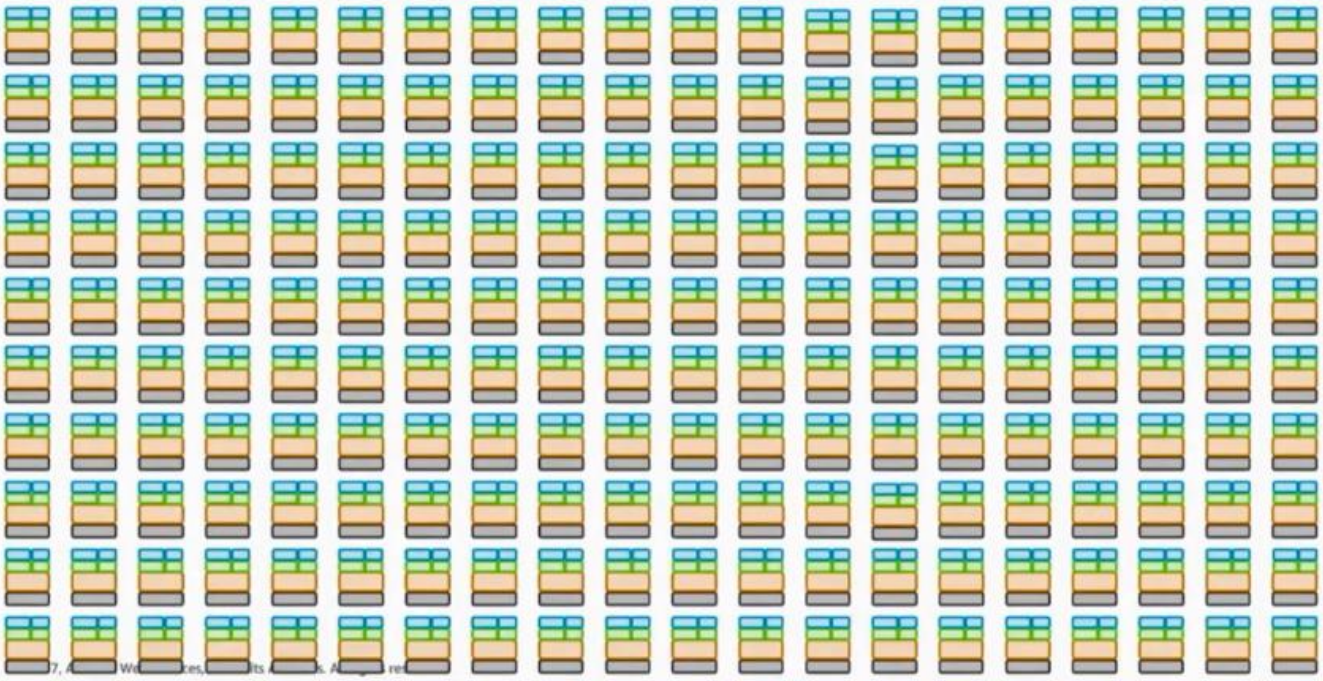


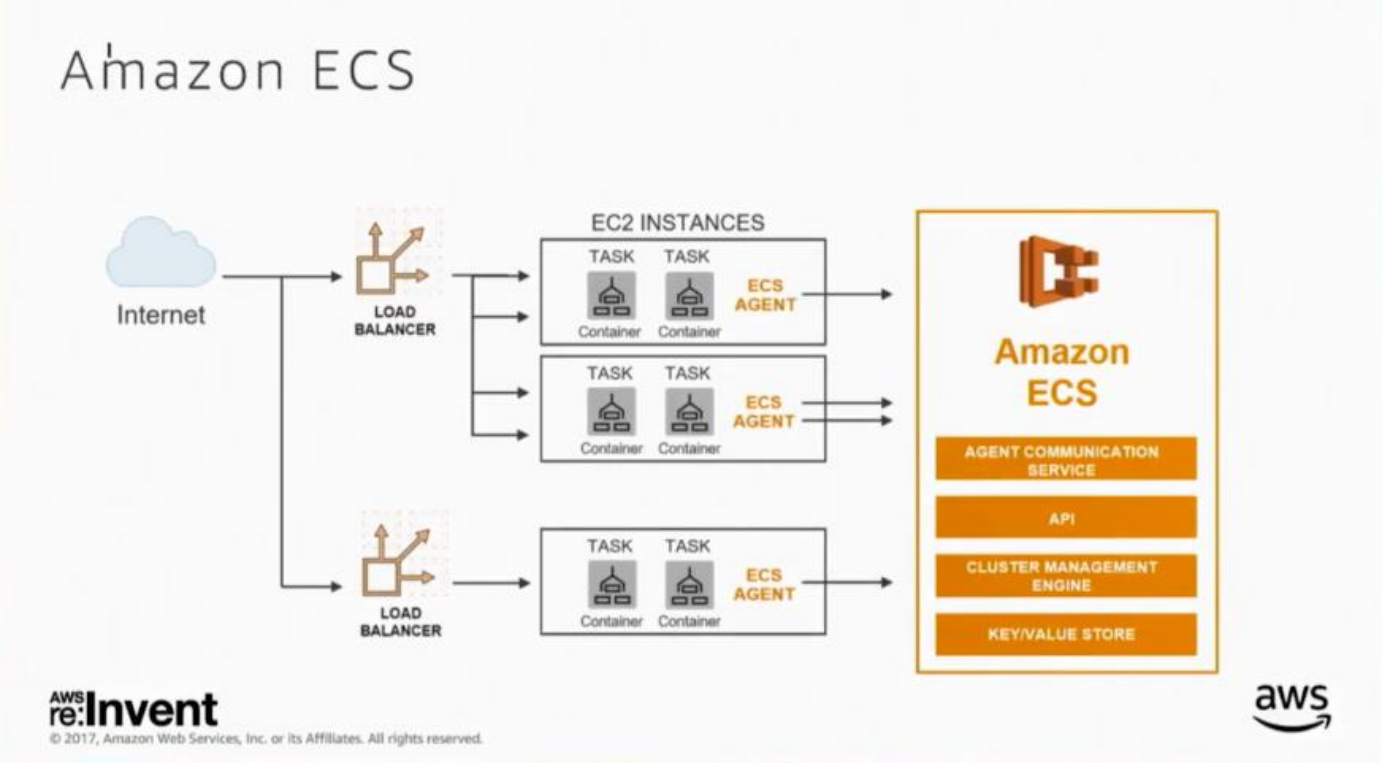
RUN

Going from a single host...



...to this is hard!



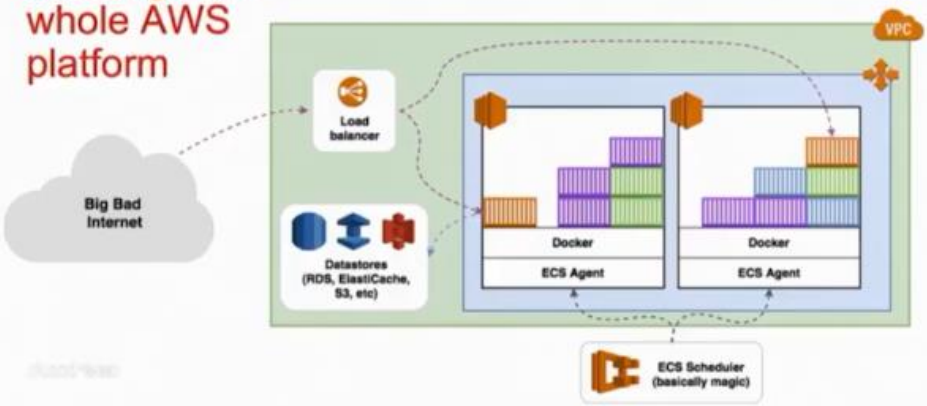


For load balancing your containers. If you have a TCP based workload that you want to run with containers you can use the NLB, if you have a HTTP based workload you can choose the ALB.

Remember...

BuzzFeed

Leverage the whole AWS platform



WeWork

Chris Tava
Josh Davis

Startups are chaos

- We had three ways to Deploy
- Security setup was super light and too open
- Lots of variation in infrastructure

After 810 days of uptime...

platform-prod-1	i-4d525f67	m3.xlarge
platform-qa-2	i-d82fd4bb	m3.large

Launch time	August 22, 2015 at 10:40:30 PM UTC-4 (19440 hours)
Termination protection	False
Lifecycle	normal
Monitoring	detailed
Alarm status	1 of 1 in No Data
Kernel ID	aki-88aa75e1
RAM disk ID	-
Placement group	-

On 6/20/17 - network card failure

Time for a big change

- Needed to be able to quickly recreate our environment
- Get it ready for [more] scale
- Secure the platform
- Adopting a devops culture
- KISS

Fieldlens - scale, consumer use

- 80K users, 110K projects
- 1.6M posts, 2.2M comments, 4.1M media
- Mobile / web clients, background jobs
- Enterprise users expect reliability!

What we started with

Docker Locally in development: docker-compose

Some micro-services

Jenkins - Build and Test, CI/CD

Environments - dev, qa, prod

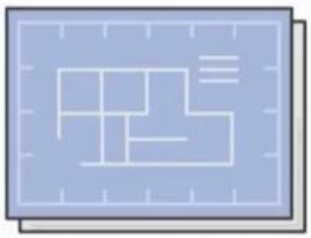
Where we want to be

Docker images for everything

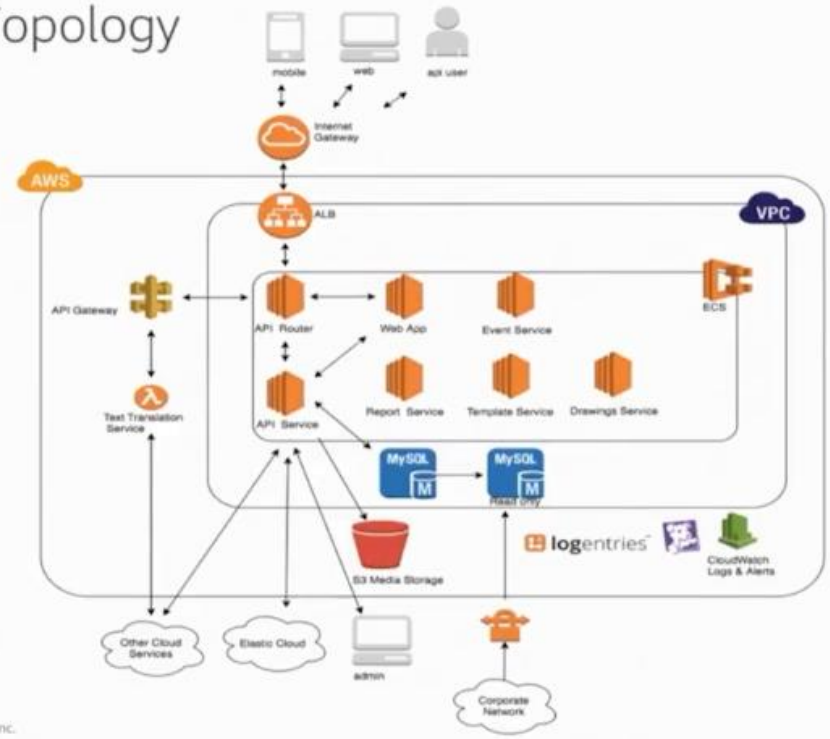
Uniform deployment / management

New micro-services are easy to create

Enhanced security / scalability / reliability



Desired Topology



wework

ECS Dev, QA, Prod

Service Name	Status
develop-weos-source-web-ecs-Service-1WKLW3K...	ACTIVE
develop-stargate-fieldlens-proxy-Service-X5A764X...	ACTIVE
develop-queue-service-ecs-Service-1E0Z7O6W36...	ACTIVE
develop-report-pdf-ecs-Service-14P7ROU2NR7TM	ACTIVE
develop-platform-ecs-Service-ZZAB078JG5DY	ACTIVE
develop-templating-service-ecs-Service-LDF7RA1I...	ACTIVE
develop-kong-ecs-Service-MCFNNEZFSAHR	ACTIVE
develop-tensorflow-service	ACTIVE
develop-mississippi-service-ecs-Service-12805LH...	ACTIVE
develop-kong-internal-ecs-Service-872TZCLK1Z3B	ACTIVE

Service Name	Status
qa-weos-source-web-ecs-Service-1RD6IJ1I6W7AG	ACTIVE
qa-drawing-service-worker-ecs-ServiceWorker-4P...	ACTIVE
qa-drawing-service-api-ecs-Service-E2UUYB41F2XX	ACTIVE
qa-templating-service-ecs-Service-5BNL4YRTXTJ4	ACTIVE
qa-hook-service-Service-1NRI2SQTTN68R	ACTIVE
qa-kong-ecs-Service-1OIDXABLHGJIH	ACTIVE
qa-report-pdf-ecs-Service-JA8GH1TMRON9	ACTIVE

Service Name	Status
production-stargate-fieldlens-pr-Service-E5608ZVI...	ACTIVE
prod-mississippi-service-ecs-Service-KNCAMYKU...	ACTIVE
prod-kong-ecs-Service-KGDBZIPLKT6R	ACTIVE
prod-hook-service-Service-TI5CHV8BIMZS	ACTIVE
prod-platform-ecs-Service-B65UYJP4DD0Q	ACTIVE
production-templating-service-ec-Service-1UFSW...	ACTIVE
production-drawing-service-api-e-Service-VFRQW...	ACTIVE
production-report-pdf-ecs-Service-WMVI9X80XF64	ACTIVE
production-drawing-service-worke-ServiceWorker-...	ACTIVE
prod-queue-service-ecs-Service-L8MW1UMN4AM0	ACTIVE
production-fieldlens-web-ecs-Service-1X3PLKBG...	ACTIVE

How do we get there?

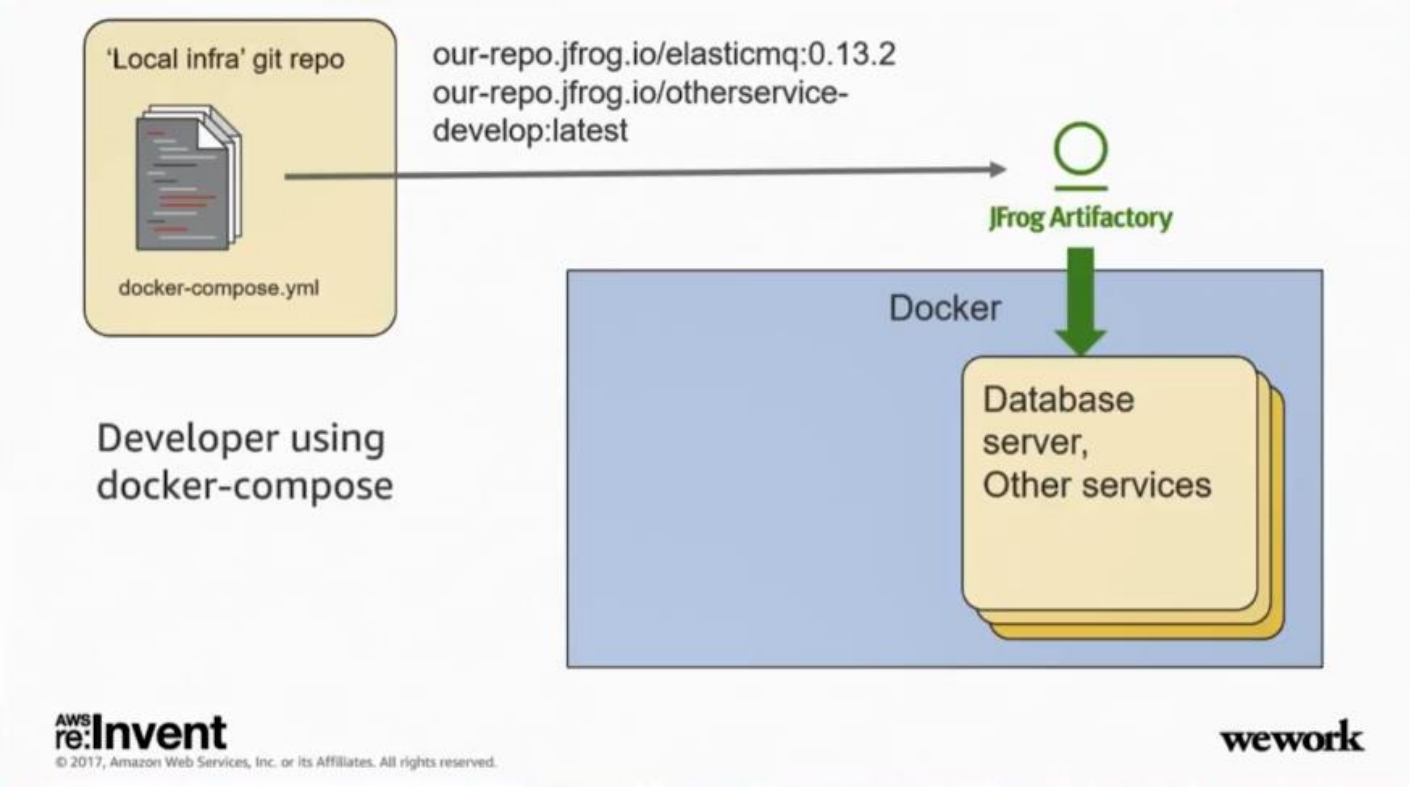
First step - Start building and publishing Docker images

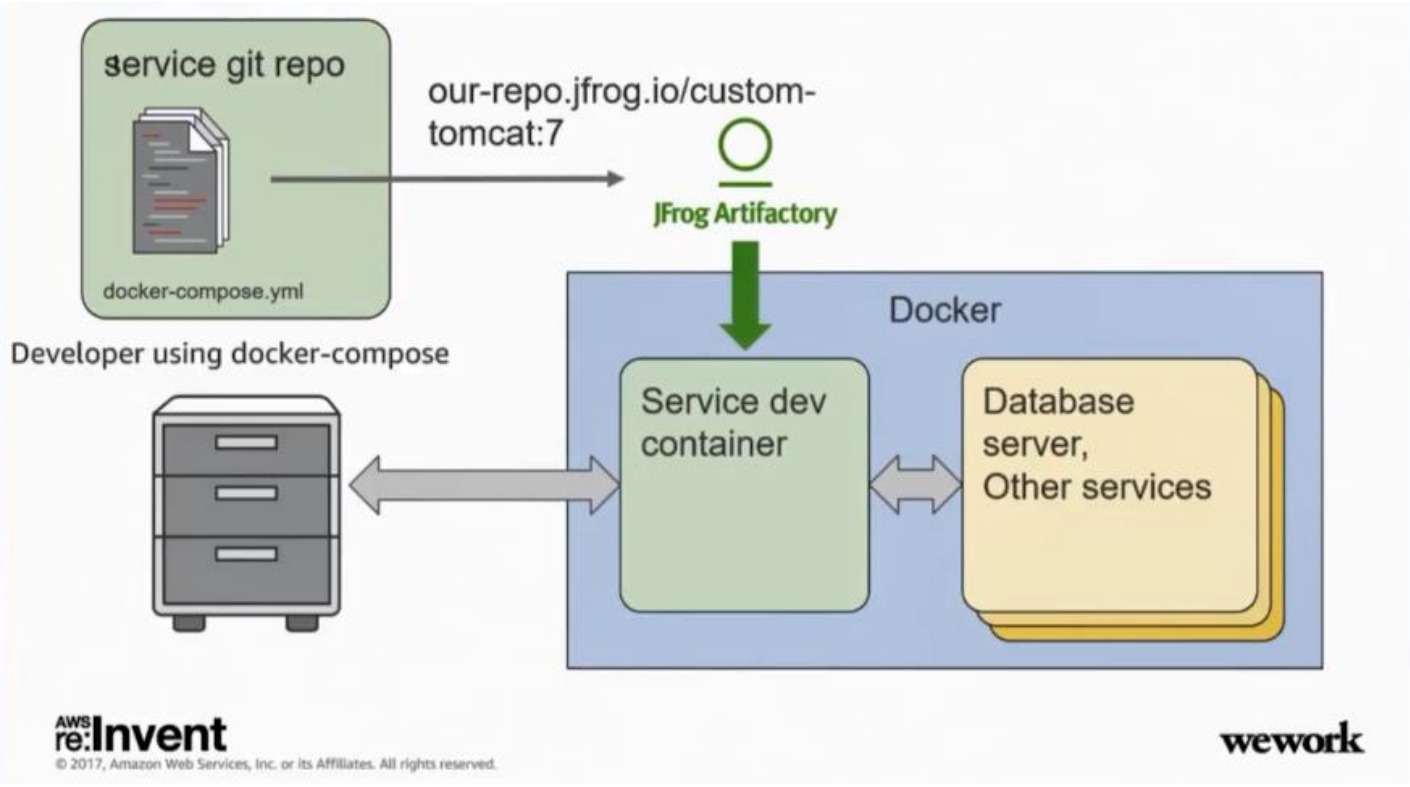
Jenkins pipelines

Build can happen outside of a container

Docker images can be deployed locally with docker-compose

Image repository, naming convention, local environment config





First milestone

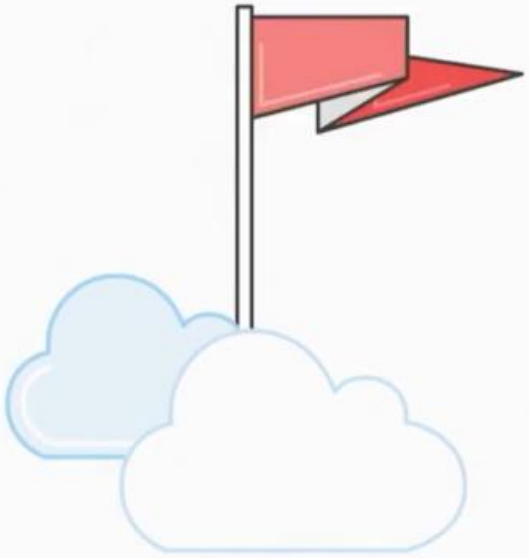
Services work in a container (locally with docker-compose)

Basic configuration established

Dockerfile / Jenkinsfile for all repos

Dependencies connected via docker network

Ready to go to the cloud!



Deploying to ECS

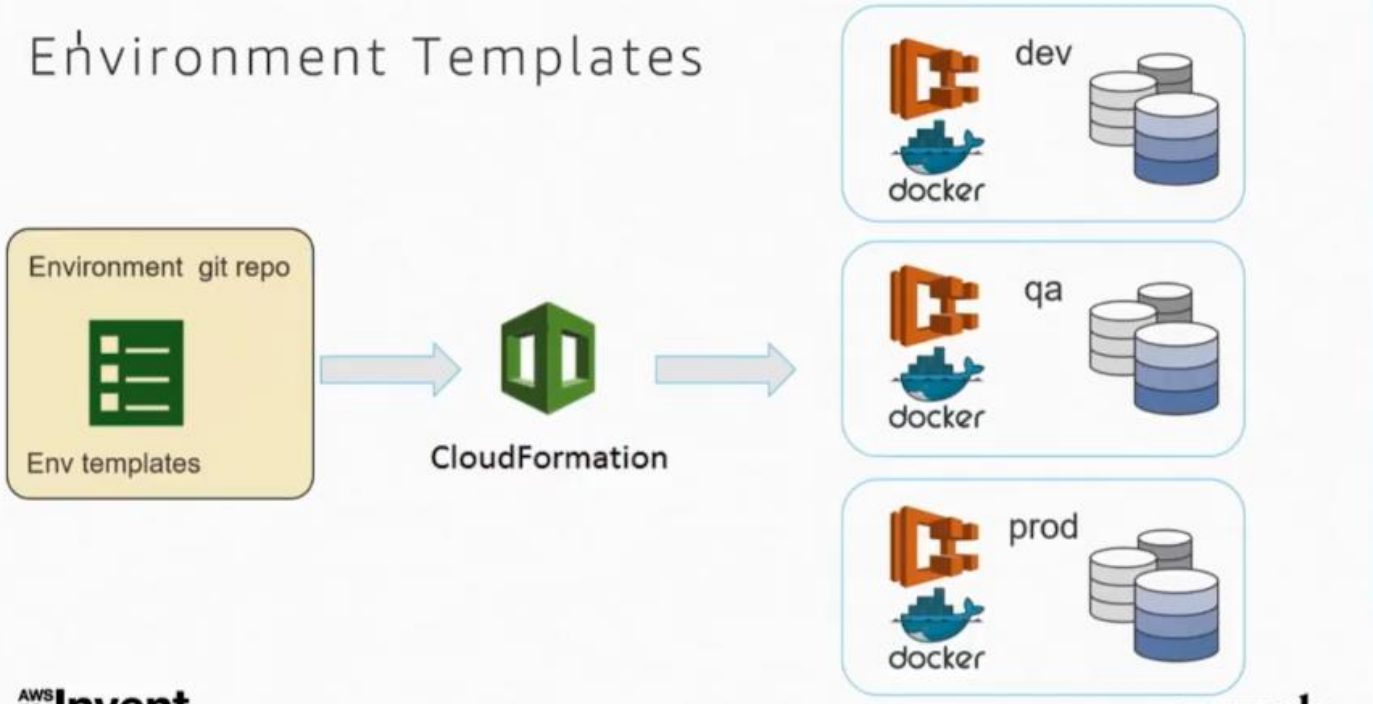
Cloud Formation - Infrastructure as Code

Environment Templates for the ECS clusters and other resources (ops)

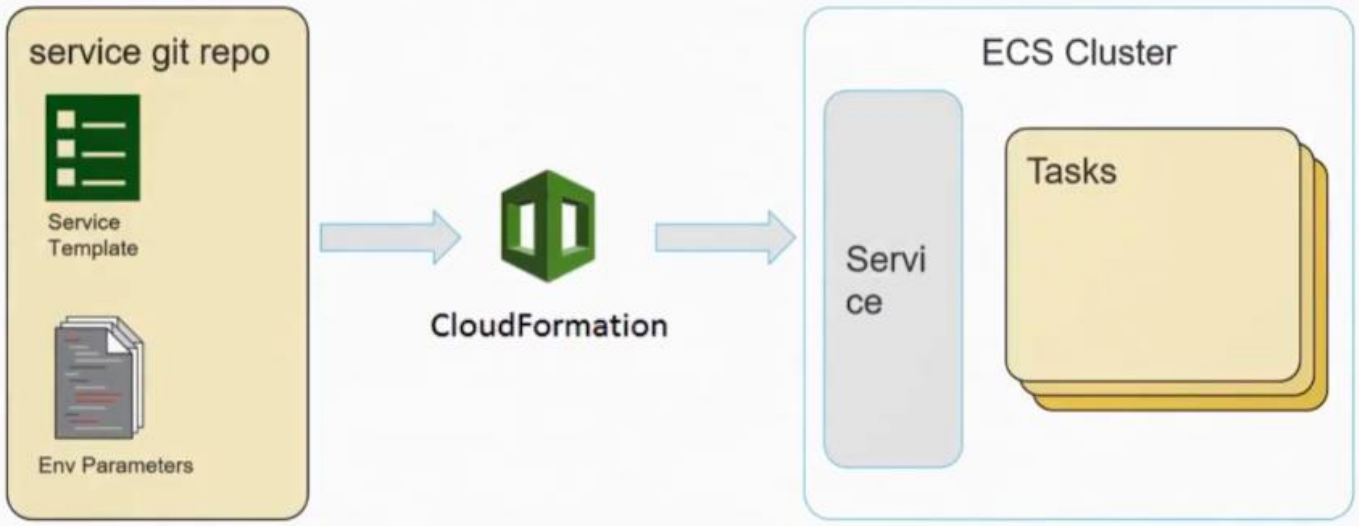
Service Templates in each repo (devs and ops)

Common shell scripts to handle basic tasks

Environment Templates

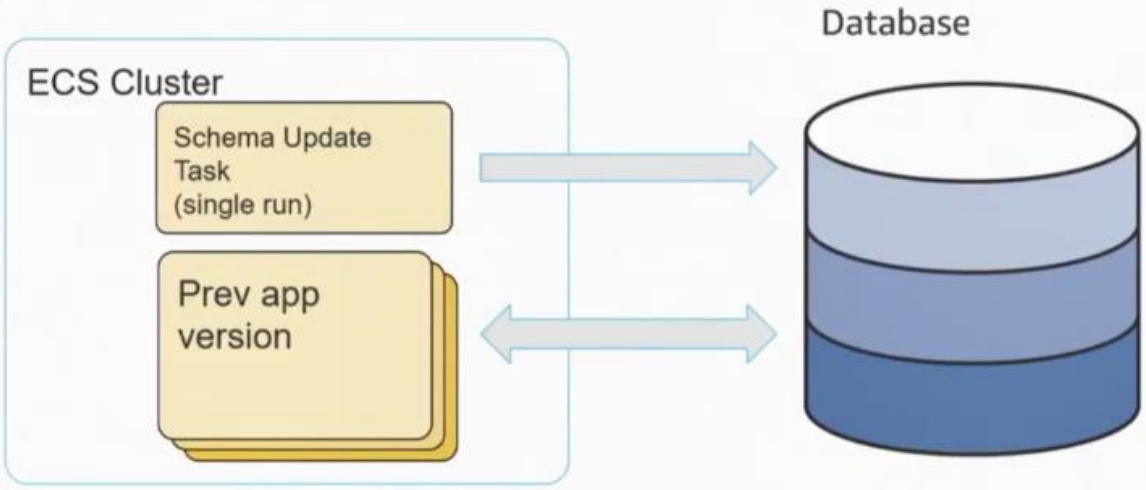


Service Templates



Database Schema Updates

Task Definition - update image with CF
Run task synchronously - Ephemeral!
Begin deployment of new code



Secrets

Encrypt sensitive config files with KMS

Developers decrypt and store encrypted configs in the service repos

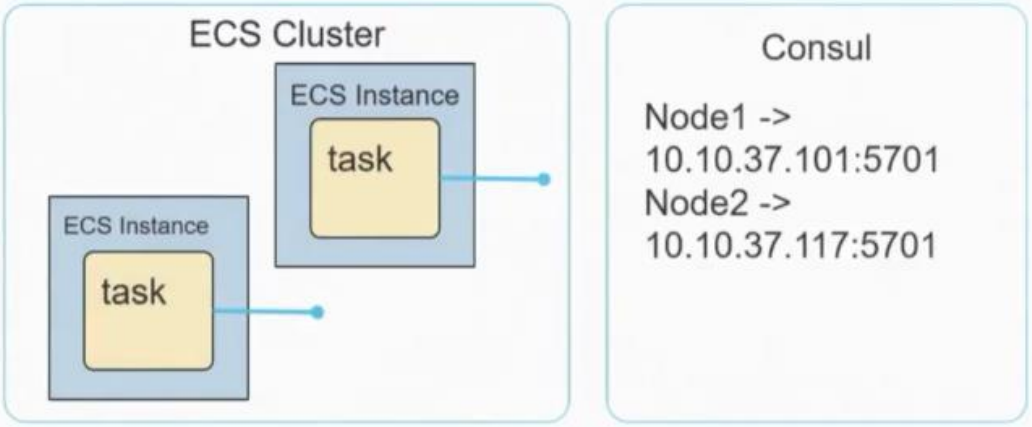
Encrypt / Decrypt privileges are controlled via IAM

Docker image contains encrypted configs

In ECS, decrypt in the entrypoint

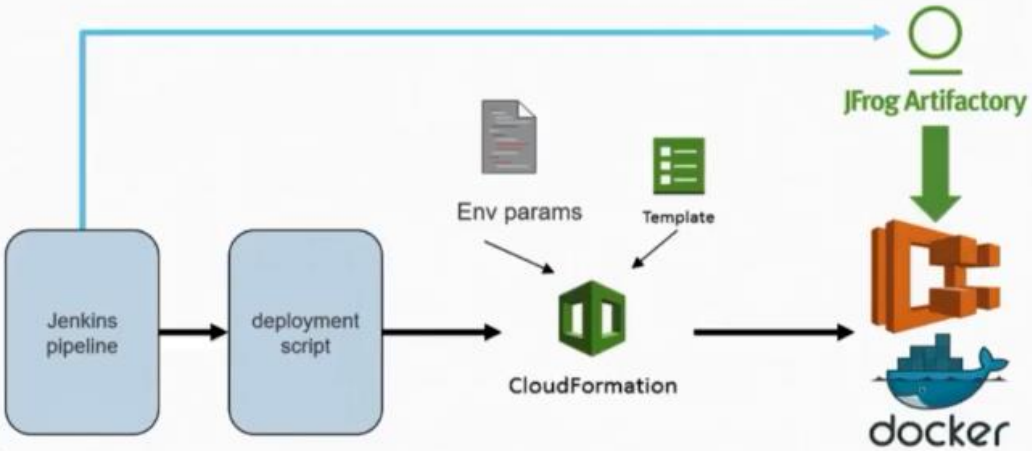
Hazelcast Cluster

Register host IP in Consul via ContainerPilot
Hazelcast start-up queries Consul for initial hosts



Deployment in Action

Deployment scripts run CloudFormation updates
CloudFormation references the docker images
ECS Scheduler rolls out the new code with zero downtime



Other Details

Entrypoint - ContainerPilot

Task start-up logging and application logging

Singleton Services

Promoting builds to environments

It's not all rainbows and unicorns...

Discovering configuration

Logentries

Datadog monitoring

Monolith needed higher ulimits

APM - TraceView

Deployment is different!

Using shell to troubleshoot - no longer a thing

Scheduler 'flapping' or 'cycling' - Rollback

Hosts constantly change - "Where are the logs?"

AWS re:Invent

wework

OpsLine

Radek Wierzbicki

Opsline

Cloud Transformation and DevOps Automation Experts

Cloud Native approach

Automate everything using CloudFormation

- VPC, security groups
- IAM and KMS
- RDS
- ECS clusters (scalable EC2 nodes)
- ECS services
- Lambda

Version control code and configuration

aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Opsline wework aws

Why ECS?

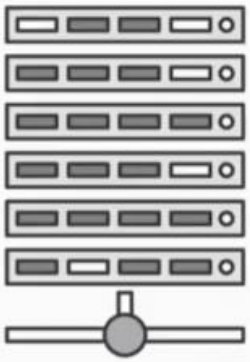
Provided by AWS
Simple
Scalable
Highly-available
Easy and fast to provision
Provides practical encapsulation of containers
Works well with CloudFormation

Amazon ECS

How did we do it?

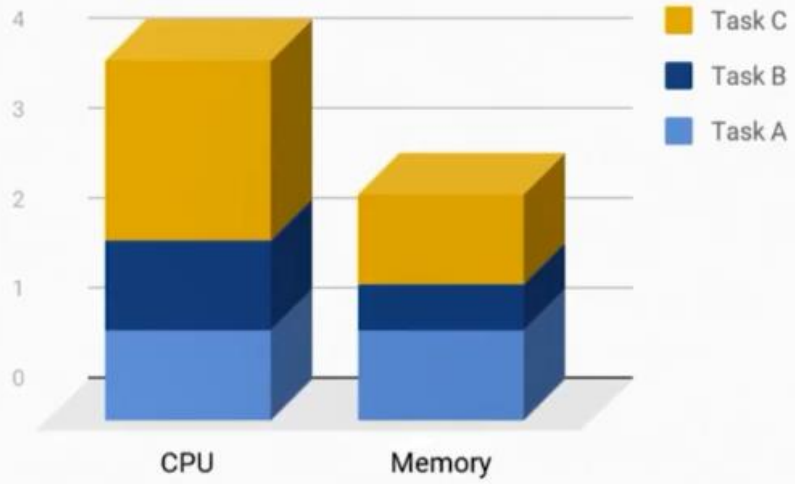
Cluster management

Provisioning - CloudFormation
Configuration management - cfn-init
Monitoring - DataDog



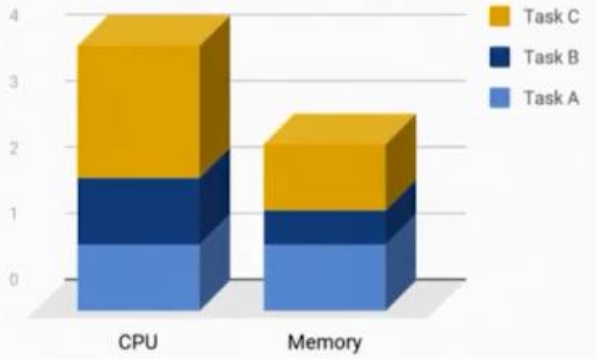
Opsline wework aws

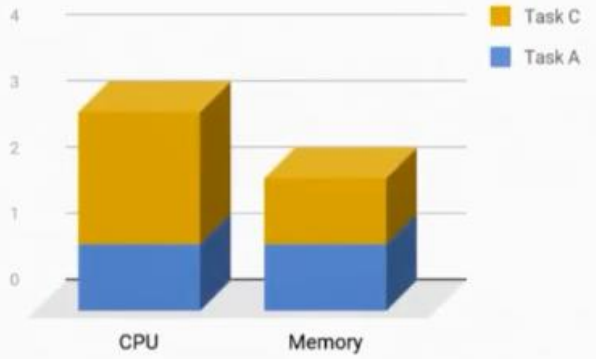
ECS cluster autoscaling



Opsline wework aws

ECS cluster autoscaling

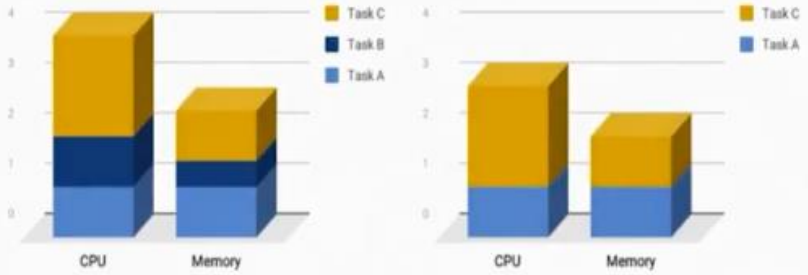




aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Opsline wework aws

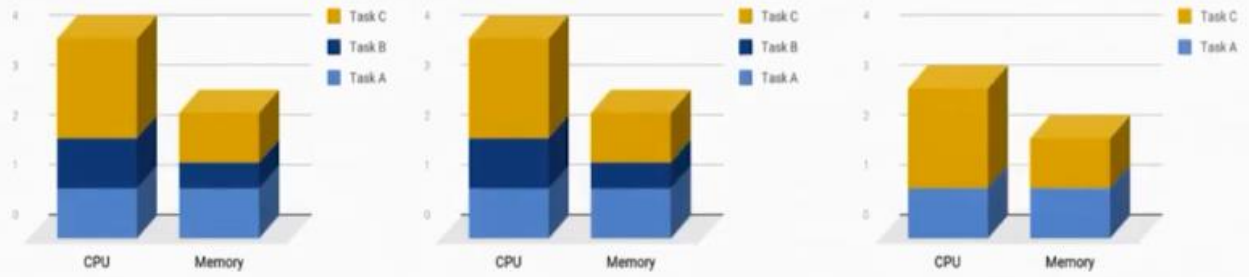
ECS cluster autoscaling



- 88% CPU utilization
- 56% Memory utilization
- Scale out when CPU is above 85%
- Scale in when CPU is below 45%

Opsline wework aws

ECS cluster autoscaling



- 92% CPU utilization
- 58% Memory utilization

aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

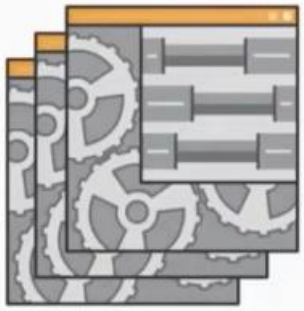
Opsline wework aws

Application management

Parametrized to run in different environments

ECS service, ECS task definition

IAM task role (access to S3, SQS, KMS, etc)



Opsline wework aws

Application monitoring

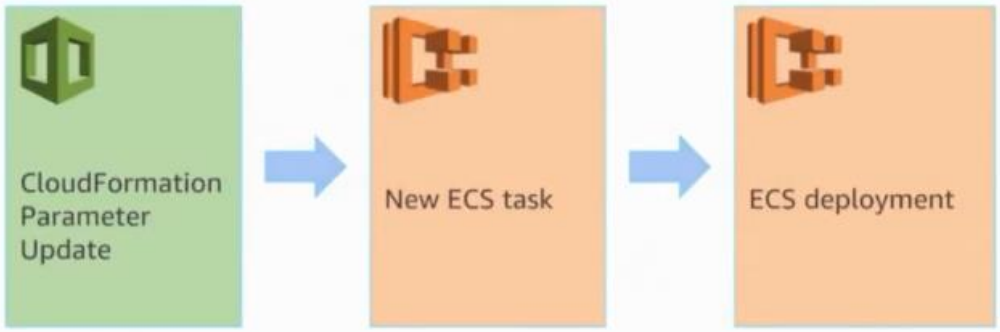
Load balancer health checks

Sidecar containers

DataDog

Opsline wework aws

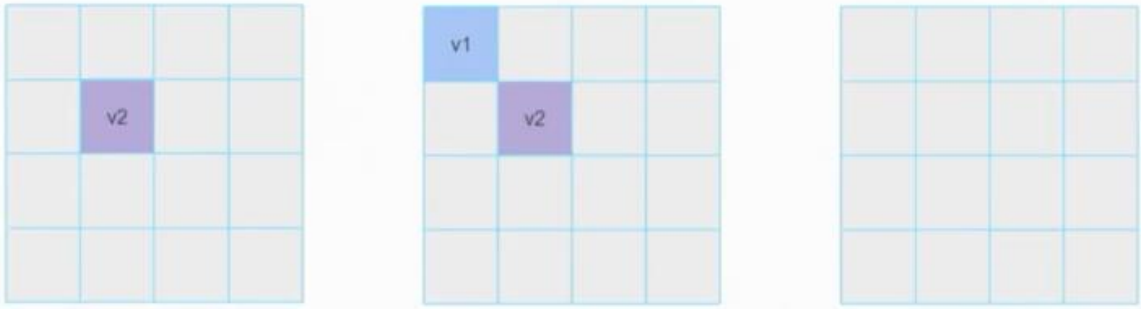
Application deployment



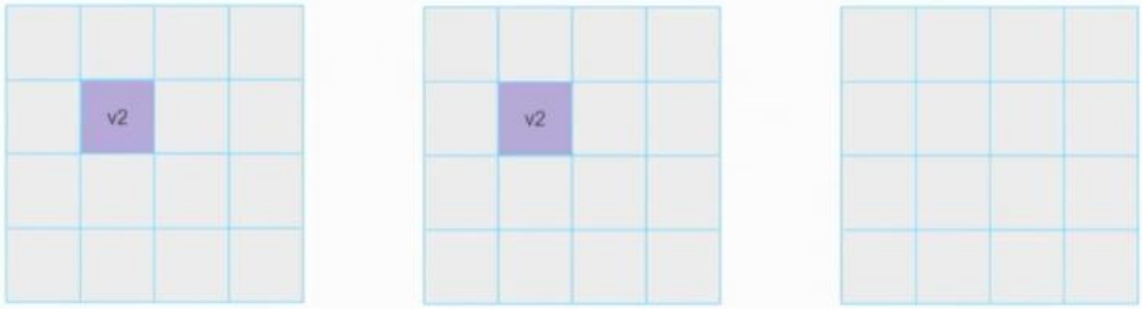
Opsline wework aws

Application deployment

Application deployment



Application deployment



Opsline wework aws

The DevOps Way

Simplicity
Visibility
- Troubleshooting
- Deployments

Opsline wework aws

The DevOps Way

Education
- Documentation
- Training

Empowerment
- Documentation and best practices
- Shell repository with templates and scripts
- Jenkins shared libraries
- Standardized Jenkins project

Collaboration

Opsline wework aws

wework

WE ARE HIRING

For more information, please contact:
ctava@wework.com

Opsline

DevOps On-Demand

For more information, please contact:
connect@opsline.com

THANK YOU!