Cloud Platform Engineering Team at Expedia Group

Subject matter areas
- Cloud Architecture
- DevSecOps
- CI/CD
- Compliance Automation

Expedia Group

I work at Expedia Partner Solutions brand, one of the many Expedia Group brands

I'm also a cross brand technology champion which means I encourage our brands to share technical knowledge with each other

Pipeline

```
= Continuous Integration &&
        (Continuous Delivery || Continuous Deployment)
```


———————-

Pipeline: the mechanism by which units of software are built, tested and delivered into test and production environments

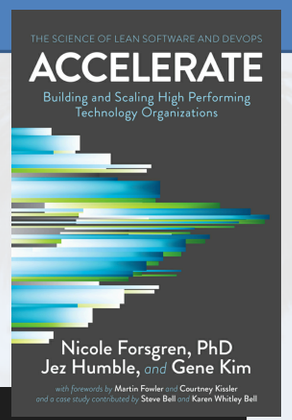
What will we cover?

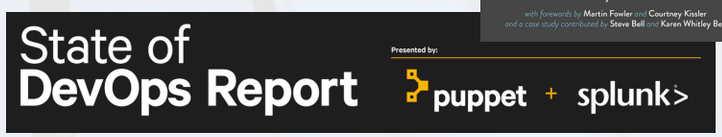
- Pipeline user behaviour
- Pipeline metrics
- Abstractions for pipeline design
- Pipelines for applications ***and infrastructure***
- Pipeline security

It’s not enough to provide our teams with a good CI/CD tools. They need to use them effectively to increase organisational performance.

State of DevOps Survey

- Jez Humble
- Gene Kim
- Nicole Forsgren
- 2013 - 2018





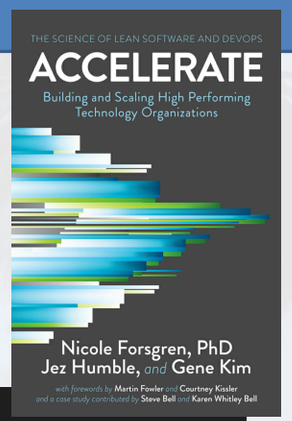
DORA = Dev Ops Research and Assessment consultancy

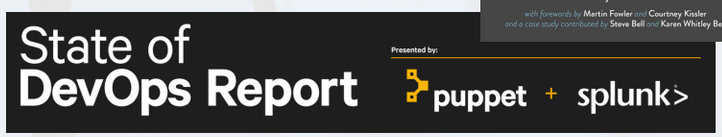
Produced own version of State of DevOps Report in 2018 and the Accelerate Book

Disclaimer: I’m not in any way affiliated with DORA, Puppet or Splunk and I do not make any money from the sale of their products or services (unfortunately!).

Likert-type scale

1. Strongly disagree
2. Disagree
3. Somewhat disagree
4. Neutral
5. Somewhat agree
6. Agree
7. Strongly agree





They used a likert-type scale to statistically analyse thousands of survey results to find out what factors cause high, medium and low organisational performance.

———————-

DORA primary research

“Our research uses latent constructs and statistical analyses to report good data— or at least provide a reasonable assurance that data is telling us what we think it’s telling us.”

Forsgren PhD, Nicole. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations (Kindle Locations 1998-2000). IT Revolution Press. Kindle Edition.

Organisational performance

High performers had 50% higher market capitalization growth over three years compared to low performers

The research found…

High vs low performers

Metric	High : Low
Deployment frequency	46 : 1
Lead time	440 : 1
MTTR	170 : 1
Change failure rate	1 : 5

They found some staggering differences in the metrics between the high and low performing groups for…

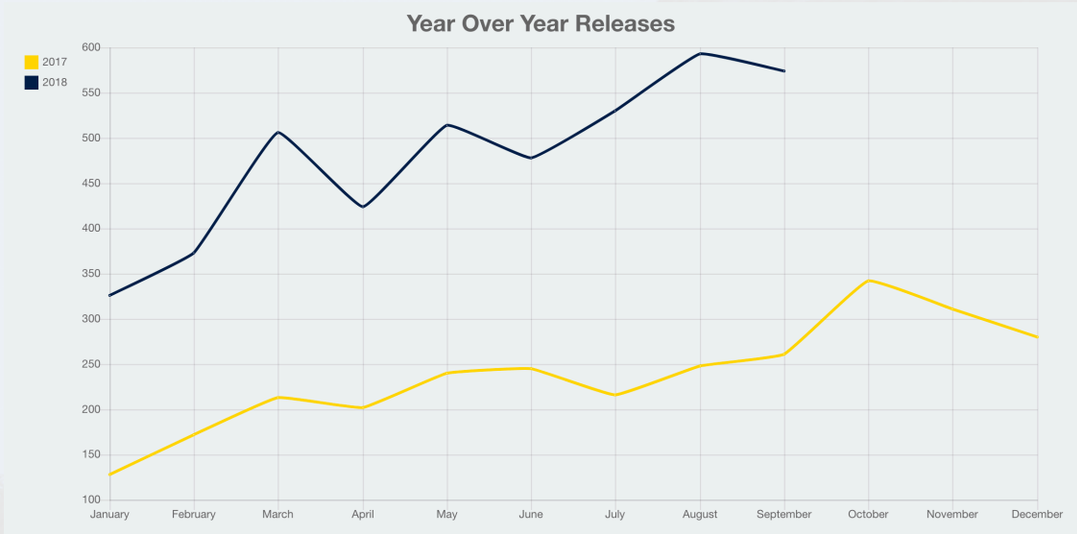
_____

Deployment frequency is used as a proxy for batch size

Lead time is the time taken to go from first code commit to the software running in production

MTTR = Mean time to recover

Expedia Partner Solutions Deploy Freq.

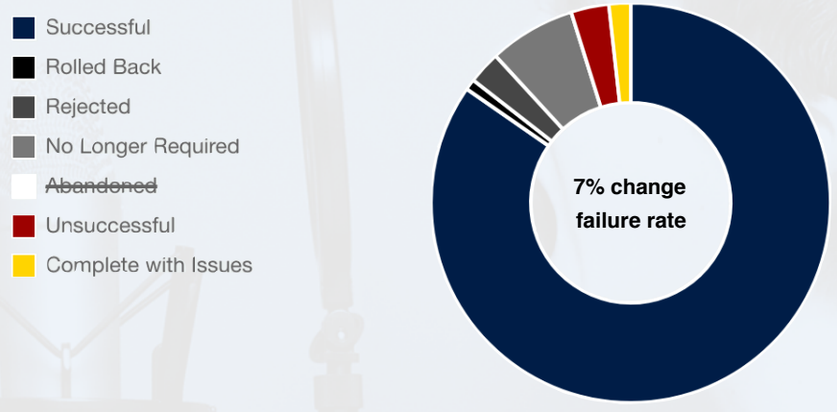


Increase due to engineering team growth and better tools and processes

Nov/Dec 17, Holiday change freeze, Las Vegas reInvent…

EPS Change Failure Rate

Last Week Releases



How can we improve the metrics?

Metric	High : Low
Deployment frequency	46 : 1
Lead time	440 : 1
MTTR	170 : 1
Change failure rate	1 : 5

How can we improve the metrics for our teams?

Forsgren, Humble and Kim

Software delivery performance is correlated with organisational investment in DevOps

Highlights the need for buy in from leadership

Forsgren, Humble and Kim

Software delivery performance is negatively correlated with deployment pain

Highlights the need to help our teams release faster and easier

Capabilities

- Continuous delivery
- Architecture
- Product and process
- Lean management and monitoring
- Cultural

We will look at CD and architecture capabilities only

_____

Focus on Capabilities because…
…leadership overestimates maturity - we're mature, no need to invest further
…practitioners are less optimistic - capabilities can always be improved

Continuous delivery capabilities

- Use of Version control
- Scripts and config more important than source code!

Continuous delivery capabilities

- Trunk based development
- Branches live less than one day

“Trunk-based development: –High performers have the shortest integration times and branch lifetimes, with branch life and integration typically lasting hours or a day.”

Forsgren PhD, Nicole. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations (Kindle Locations 2770-2771). IT Revolution Press. Kindle Edition.

Continuous delivery capabilities

- “Shift left” on security
  - InfoSec review early design work
  - Tools: eg, Fortify, Application libs and templates
  - Training, eg OWASP

“When building security into software is part of the daily work of developers, and when infosec teams provide tools, training, and support to make it easy for developers to do the right thing, delivery performance gets better.”

Forsgren PhD, Nicole. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations (Kindle Locations 1172-1174). IT Revolution Press. Kindle Edition.

OWASP = Open Web Application Security Project

Architecture capabilities

- Use a loosely coupled architecture
- Services can be tested in isolation

Architecture capabilities

- Teams are empowered to choose their tools

If teams can choose tools, we have to provide tools that are so helpful that the teams will want to use them.

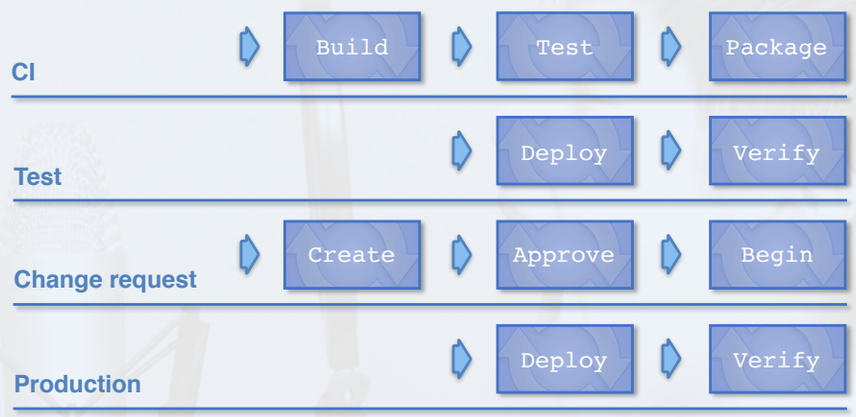
“good” rather than “right” - Find the ones that work for your organisation

Early stage start up vs Enterprise

What are the problems we need to solve?
- Deployment approvals, compliance, regulation, many teams
- It is not uncommon to find a proliferation of similar but not identical CI/CD pipelines. When a requirement changes, how do we propagate the change to all pipelines? How do we ensure each pipeline is following best practices for security, compliance or regulatory requirements? How do we avoid the pipeline becoming a maintenance burden? We want our pipelines to deliver value. Ideally we want our engineers to spend more time thinking about what goes through the pipeline than they do thinking about how it goes through the pipeline. To do this we need to optimise for developer experience.

Pipeline responsibilities
1. What to build
2. How to build
3. Where to store built artefacts
4. How to test: static, security, unit, functional, non-functional etc
5. Approval process
6. How to deploy built artifacts
7. Where to deploy built artifacts - eg test, multi region prod
8. CD: Verify deployments
9. Generate metrics
10. Generate notifications

Traditional CI/CD

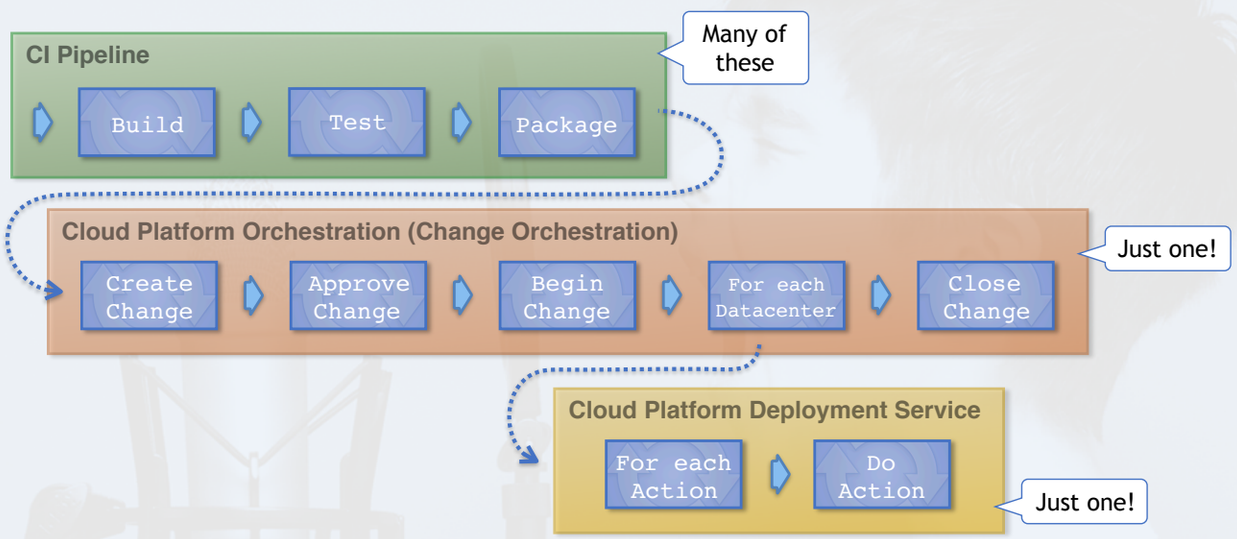


Using a single pipeline for all steps from code commit to validated in production requires repetitive config. The repetition is seen within a single pipeline instance and across teams that are using similar pipelines.

Not good for infrastructure deploys because whole pipeline ends up with infrastructure CRUD permissions.

DRY!!

Multiple specialised pipelines



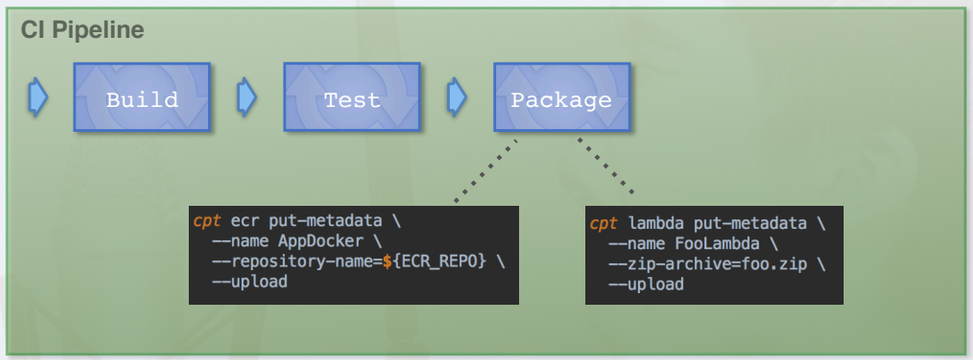
We've not moved to a model that looks like this…
- Traditional CI tool for build, test, package

Then we start the CP Orchestration pipeline. Naming is hard!
- We only need one of these, because the only thing that changes between pipelines is which datacenter we need to deploy to.

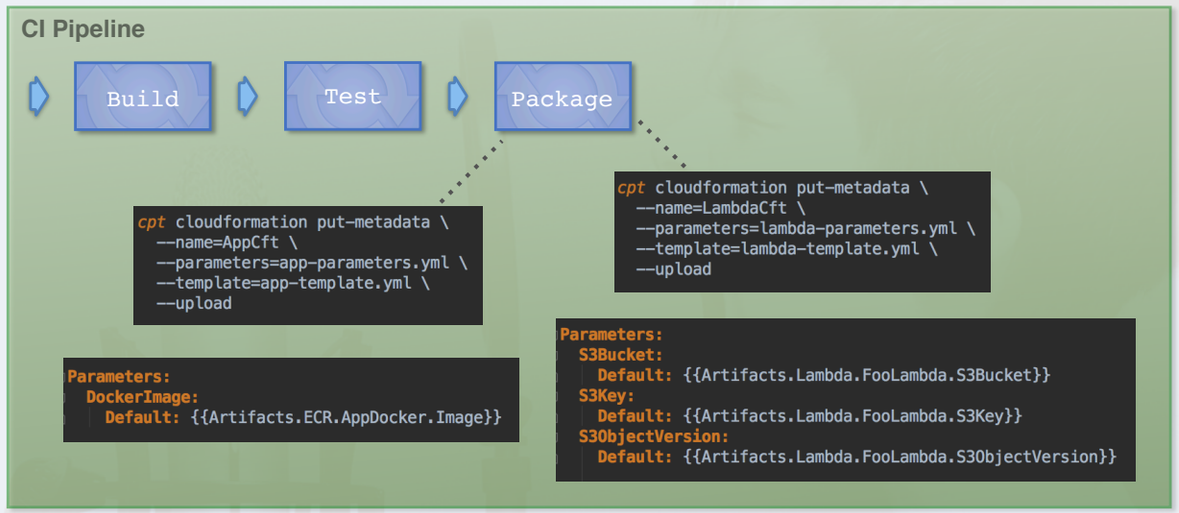
The deployment for each datacenter is done by starting the Cloud Platform Deployment Service pipeline to perform each deployment.
- Again, we only need one of these…

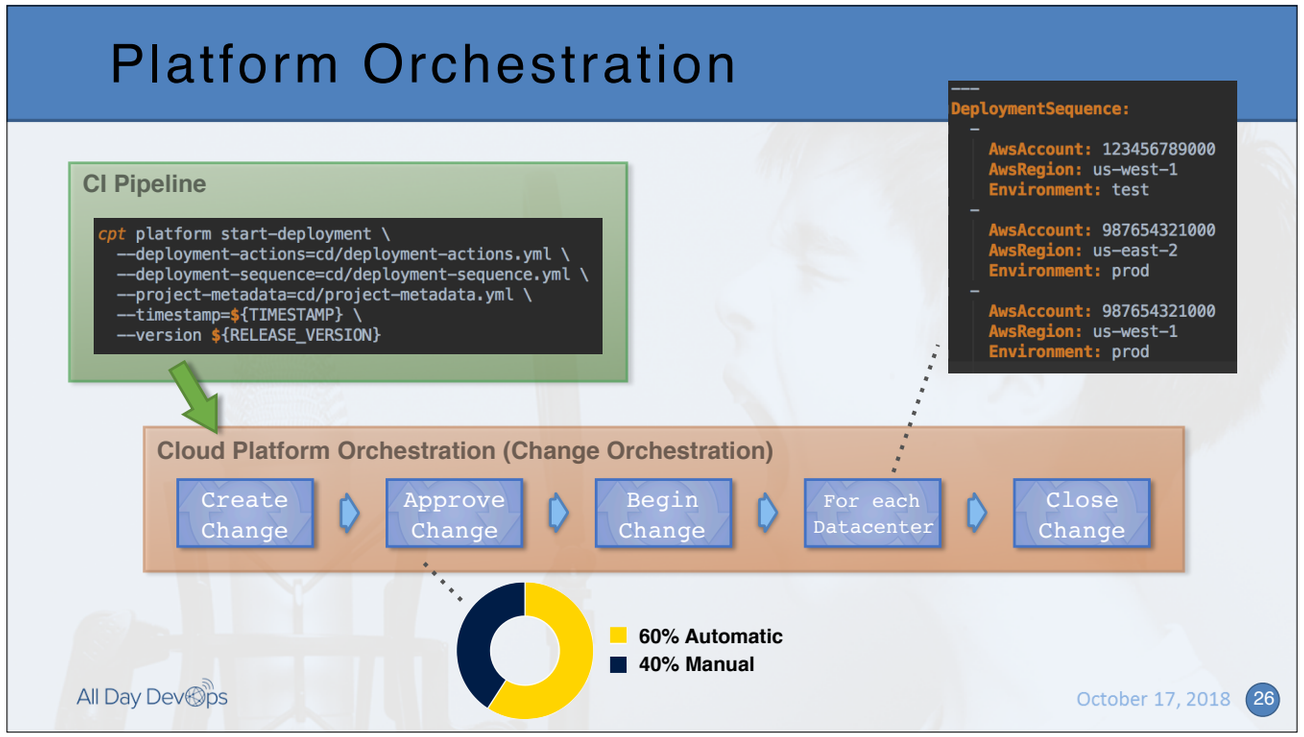
At EPS we're AWS Step Functions for these pipelines. Step Functions are server-less distributed state machines. These are smart pipelines on steroids! This server-less approach means we pay minimal costs while pipelines are running. We also pay nothing when pipelines are not running. AWS allow us to run up to 1M executions of the pipeline concurrently so it's not a problem to have only one instance.

Build deployable artifacts



Parameter value look-ups





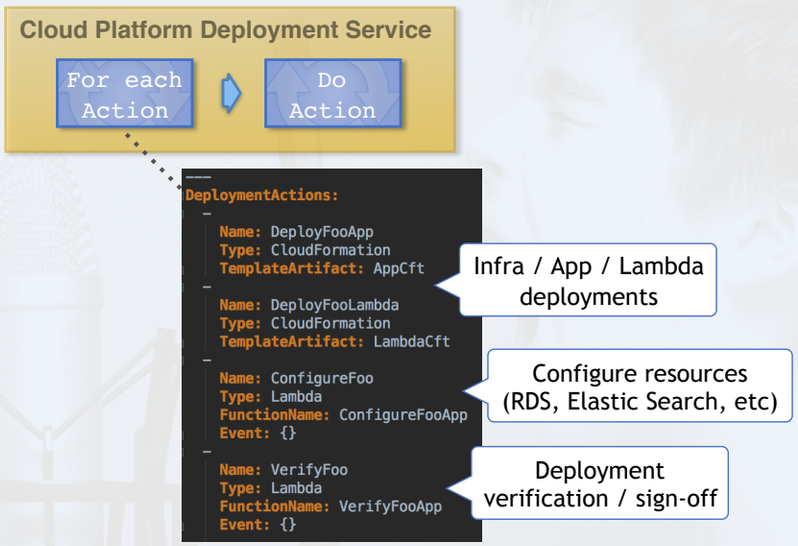
Platform Orchestration pipeline features

- Some changes are automatically approved
- Quality assurance checks for prod deployments
    - Change is properly documented?
    - Pull request for the change?

———-—-

A single pipeline is used for all changes

Deployment action types



Deployment actions are processed sequentially. We currently support two types of action…

_____

A single pipeline instance is used for all deploys

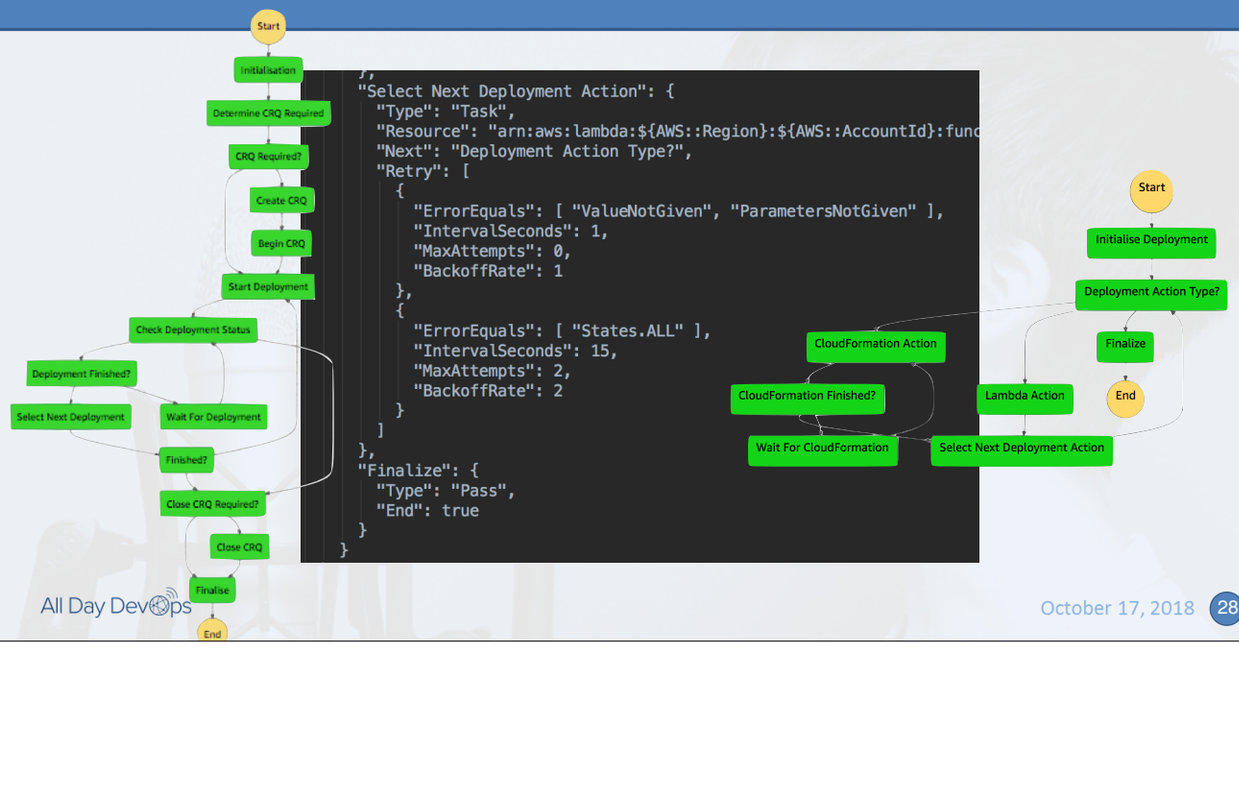
We plan to add more deployment actions types next year, eg
- actions to deploy containers and Lambdas without using CloudFormation template properties
- actions to invoke other pipelines, eg Blue/Green traffic switch pipeline

At EPS we want to empower our developers so we let them write the IAM policies that will be used to execute their code. How do we ensure that they follow the ‘principal of least privilege and choose the minimum permissions necessary for their code execution? We believe our people want to do the right thing once they know what that is. We’re currently looking at some tooling to help with this. cfn_nag and cfn_lint can be used to identify overly permissive IAM policies and Security Groups at build time.

https://github.com/awslabs/cfn-python-lint

https://github.com/stelligent/cfn_nag

State Machines

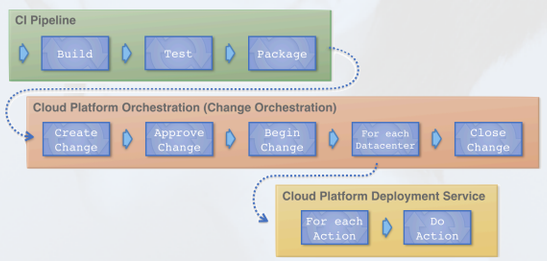


This is how the state machines look in the console. We can watch the state transitions happen as they execute which is nice.

The state transitions are defined using Amazon States Language. Using this technology means we've been able to declaratively define error handling, retries and exponential backoff when Amazon API calls get throttled, which is great as we've not had to write our own code for doing that. We have implemented our pipeline states using AWS Lambda Functions in written in Python.

Specialised pipeline advantages

- Modular design
- Abstract by responsibility
  - Choose best tool for each job
- Principal of least privilege
- Reduced attack vectors



Modularity: Easy to swap components

Abstract by responsibility: Best tools for: build, compliance, provisioning and deployment etc

Principal of least privilege: Don’t have to give infrastructure CRUD permissions to the whole pipeline.

Serverless pipelines can run in accounts with no EC2 and no VPC. This means attack vectors are limited to AWS public API endpoints. AWS takes care of securing those for us.

Dependency resolution

- Do it at build time…

  …for reliable deployment and autoscaling
- Self contained deployables
- Use containers for build tools
- Use content digests

Resolve dependencies at build time not deploy time
- doing this makes deployment, rollbacks and auto/manual scaling highly reliable
- means we can make version pinning the exception not the rule so tech debt does not build up

Do this by storing the resolved dependencies in self contained deployables: Containers, Machine images etc
- At deployment time we only need to resolve: self contained deployable; environment config and secrets

Provide build tools in containers for portability and build agent reliability
- developers can run tools on laptop without complex dependencies (only need to install Docker)
- runs in CI the same as it did on laptop
- no complex CI tool set up
- files can be copied out of the container afterwards if necessary
- optimise Dockerfiles: install dependencies first, add your own code last

Uses content digests for storing all artifacts
- Only deploy what has changed
- Save storage space

Future work

- Blue/green traffic switching
- Application templates
- More metrics
- Inner/open source
- Plugin architecture

Plugin architecture for command line tools and CI/CD services

Code examples

Summary

- Invest in DevOps
- Help teams increase performance
- DRY - Use specialised pipelines
- Principal of least privilege (infra deploys)
- Optimise use of dependencies

Thank You All Day DevOps Sponsors

Platinum Sponsors

Sonatype

cloudbees

Gold Sponsors

puppet

GitLab

XebiaLabs
Enterprise DevOps

General Dynamics
Information Technology

cisco DEVNET
developer.cisco.com

Carnegie Mellon University
Software Engineering Institute

CHEF

SCALED AGILE

Media Sponsors

TechBeacon

DevOps.com

DZone

Solutions Review

Thank You All Day DevOps Supporters

If you don’t get time to reach me on Slack in the next hour, then please reach out to me on Twitter @techabstraction