codesushi

DON'T LOSE YOUR HEAD IN THE CLOUD

AWS

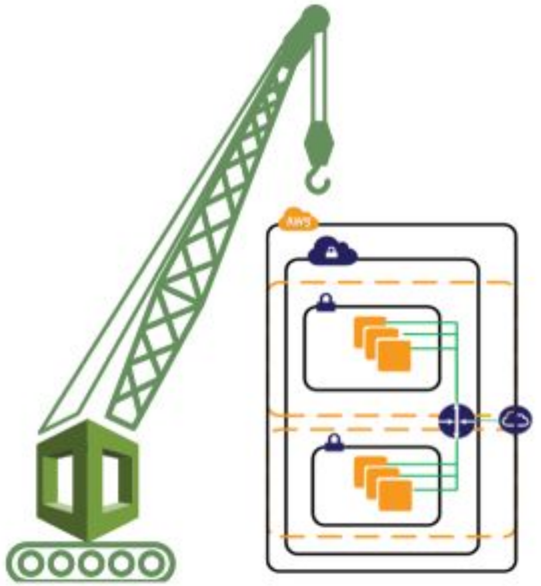
Cloud Formation

Andrew Maleszewski

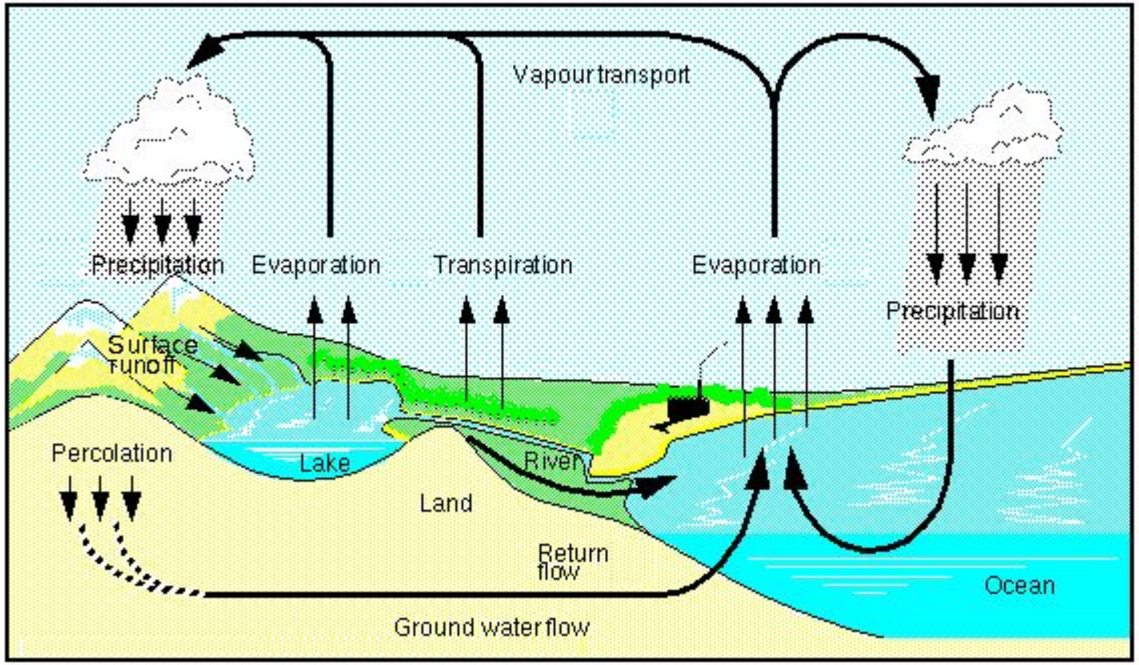
codesushi

Agenda

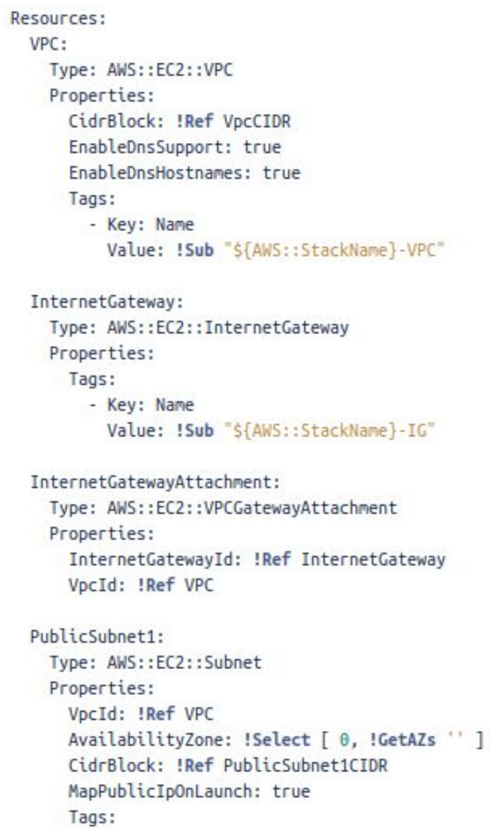
- What's Cloud Formation
- What does it offer
- How did we use it
- How can you start using it



codesushi



Well yes, but actually no

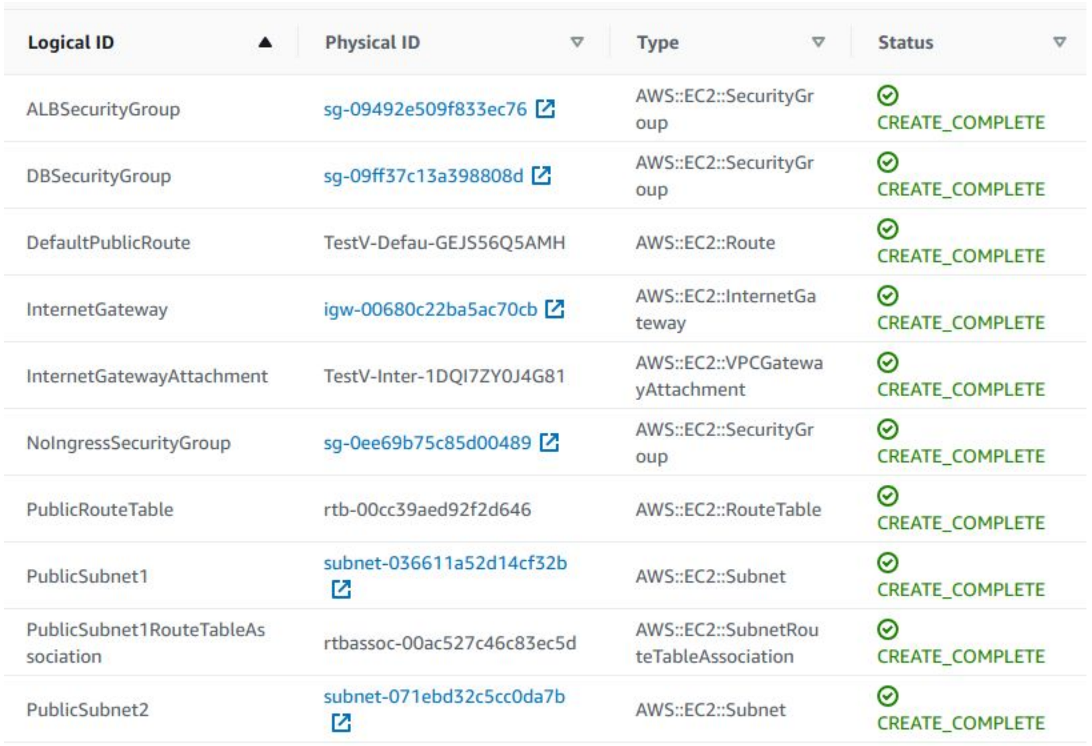


```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Sub "${AWS::StackName}-VPC"

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Sub "${AWS::StackName}-IG"

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
```



Logical ID	Physical ID	Type	Status
ALBSecurityGroup	sg-09492e509f833ec76	AWS::EC2::SecurityGroup	✓ CREATE_COMPLETE
DBSecurityGroup	sg-09ff37c13a398808d	AWS::EC2::SecurityGroup	✓ CREATE_COMPLETE
DefaultPublicRoute	TestV-Defau-GEJS56Q5AMH	AWS::EC2::Route	✓ CREATE_COMPLETE
InternetGateway	igw-00680c22ba5ac70cb	AWS::EC2::InternetGateway	✓ CREATE_COMPLETE
InternetGatewayAttachment	TestV-Inter-1DQI7ZY0J4G81	AWS::EC2::VPCGatewayAttachment	✓ CREATE_COMPLETE
NoIngressSecurityGroup	sg-0ee69b75c85d00489	AWS::EC2::SecurityGroup	✓ CREATE_COMPLETE
PublicRouteTable	rtb-00cc39aed92f2d646	AWS::EC2::RouteTable	✓ CREATE_COMPLETE
PublicSubnet1	subnet-036611a52d14cf32b	AWS::EC2::Subnet	✓ CREATE_COMPLETE
PublicSubnet1RouteTableAssociation	rtbassoc-00ac527c46c83ec5d	AWS::EC2::SubnetRouteTableAssociation	✓ CREATE_COMPLETE
PublicSubnet2	subnet-071ebd32c5cc0da7b	AWS::EC2::Subnet	✓ CREATE_COMPLETE

That’s more like it…

What does it do

- Uses templates to describe AWS infrastructure
- The template creates a Stack
- Replicability
- Reducing errors

- Infrastructure as a code

codesushi

What's supported

- ASK
- AmazonMQ
- Amplify Console
- API Gateway
- API Gateway V2
- Application Auto Scaling
- App Mesh
- AppStream 2.0
- AppSync
- Athena
- AWS Auto Scaling
- Amazon EC2 Auto Scaling
- AWS Backup
- AWS Batch
- AppStream 2.0

- AppSync
- Athena
- AWS Auto Scaling
- Amazon EC2 Auto Scaling
- AWS Backup
- AWS Batch
- AWS Budgets
- Certificate Manager
- AWS Cloud9
- CloudFormation
- CloudFront
- AWS Cloud Map
- CloudTrail
- CloudWatch
- CloudWatch Logs

- Amazon EventBridge
- CodeBuild
- CodeCommit
- CodeDeploy
- CodePipeline
- CodeStar
- Amazon Cognito
- Config
- AWS Data Pipeline
- DAX
- Directory Service
- DLM
- DMS
- Amazon DocumentDB
- DynamoDB

- EC2
- Amazon ECR
- ECS
- EFS
- EKS
- ElastiCache
- Elasticsearch
- Elastic Beanstalk
- Elastic Load Balancing
- ElasticLoadBalancingV2
- Amazon EMR
- FSx
- GameLift
- AWS Glue
- GuardDuty
- IAM
- Inspector
- IoT
- IoT1Click
- IoT Analytics
- IoTEvents
- AWS IoT Greengrass
- AWS IoT Things Graph
- Amazon Kinesis
- KinesisAnalytics
- Amazon Kinesis Data Analytics V2
- Amazon Kinesis Data Firehose
- KMS
- Amazon SimpleDB
- Amazon SNS
- Amazon SQS
- Step Functions
- Systems Manager
- AWS SFTP
- WAF
- WAF Regional
- WorkSpaces
- Shared
- Property Types
- LakeFormation
- Lambda
- ManagedBlockchain
- MediaLive
- MediaStore
- MSK
- Amazon Neptune
- OpsWorks
- OpsWorks-CM
- Pinpoint
- PinpointEmail
- RAM
- RDS
- Amazon Redshift
- RoboMaker
- Route 53
- Route 53 Resolver
- Amazon S3
- Amazon SageMaker
- Secrets Manager
- Service Catalog
- SecurityHub
- SES

codesushi

Quite a lot?

codesushi

Should we use it?

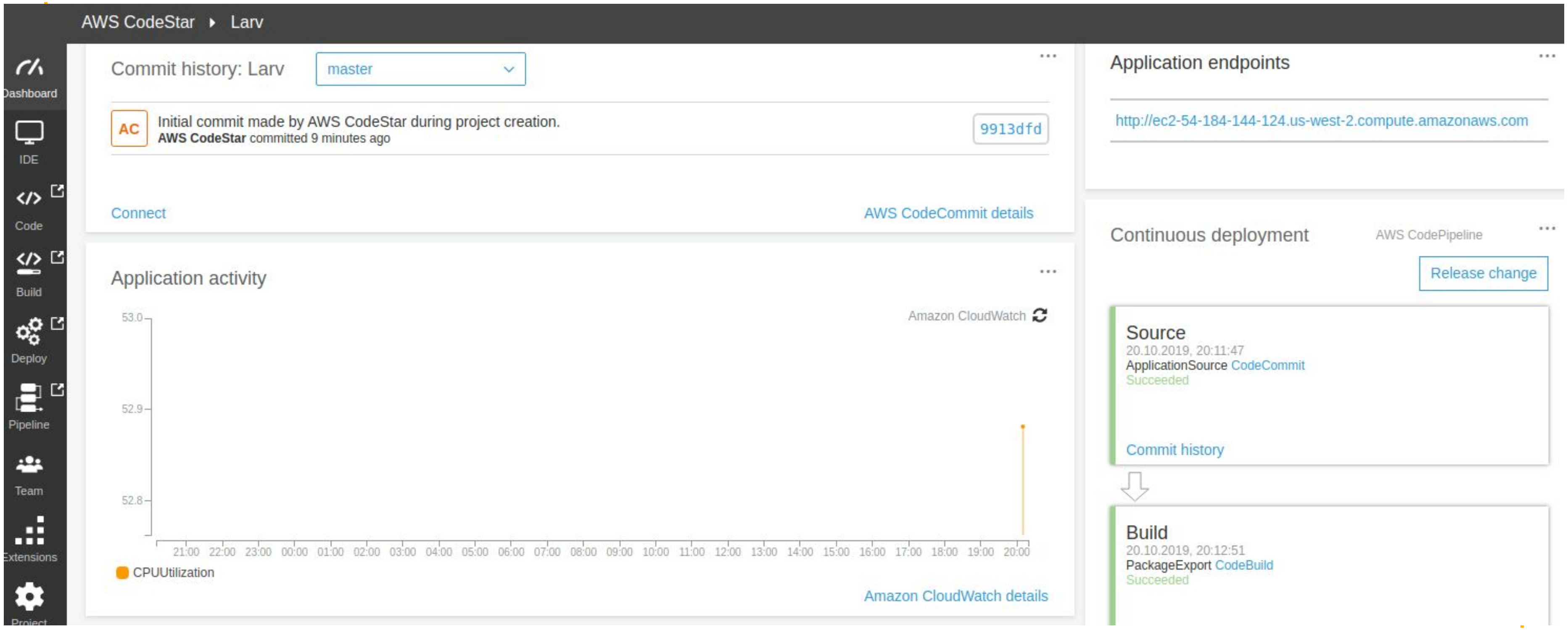
codesushi

Sure

Have I mentioned it's free?

Familiar services

- Beanstalk
- CodeStar
- AWS Amplify



AWS CodeStar ▸ Larv

Commit history: Larv — master

AC Initial commit made by AWS CodeStar during project creation.
AWS CodeStar committed 9 minutes ago — 9913dfd

Connect — AWS CodeCommit details

Application activity

Amazon CloudWatch

🟧 CPUUtilization

Amazon CloudWatch details

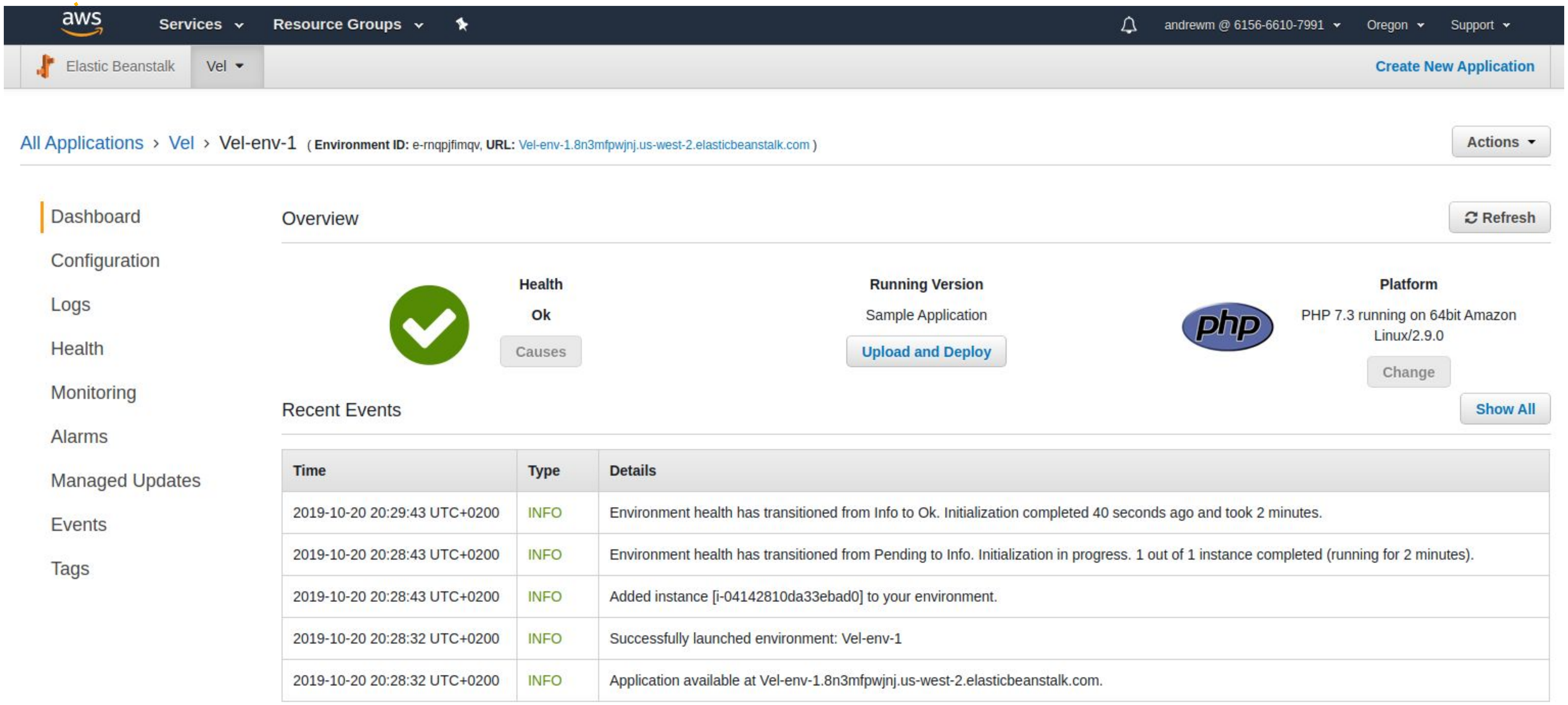
Application endpoints

http://ec2-54-184-144-124.us-west-2.compute.amazonaws.com

Continuous deployment — AWS CodePipeline — Release change

Source
20.10.2019, 20:11:47
ApplicationSource CodeCommit
Succeeded
Commit history

Build
20.10.2019, 20:12:51
PackageExport CodeBuild
Succeeded



All Applications › Vel › Vel-env-1 (**Environment ID:** e-rnqpjfimqv, **URL:** Vel-env-1.8n3mfpwjnj.us-west-2.elasticbeanstalk.com)

Overview

Health

Ok

Running Version

Sample Application

Platform

PHP 7.3 running on 64bit Amazon Linux/2.9.0

Recent Events

Time	Type	Details
2019-10-20 20:29:43 UTC+0200	INFO	Environment health has transitioned from Info to Ok. Initialization completed 40 seconds ago and took 2 minutes.
2019-10-20 20:28:43 UTC+0200	INFO	Environment health has transitioned from Pending to Info. Initialization in progress. 1 out of 1 instance completed (running for 2 minutes).
2019-10-20 20:28:43 UTC+0200	INFO	Added instance [i-04142810da33ebad0] to your environment.
2019-10-20 20:28:32 UTC+0200	INFO	Successfully launched environment: Vel-env-1
2019-10-20 20:28:32 UTC+0200	INFO	Application available at Vel-env-1.8n3mfpwjnj.us-west-2.elasticbeanstalk.com.

Familiar services

- They all utilize CloudFormation
- **Less / More** customizability

codesushi

In return

- Automation and replicability
- Infrastructure rollbacks
- Easy and clean management
- Gathering Services in custom groups

codesushi

Feature Environments

- New environment for each feature
- Create new for each backend consuming team
- No more conflicts
- Unify - PreProduction

Backend consumers

- Multiple teams / apps (*web, mobile*)
- Sometimes working in different timezones
- Sharing the same space can cause conflict
- Give everybody their own data set (*fixtures, sql dumps*)

Resolve and unify

- Expanded flow:
  - Dev -> Stage -> Prod

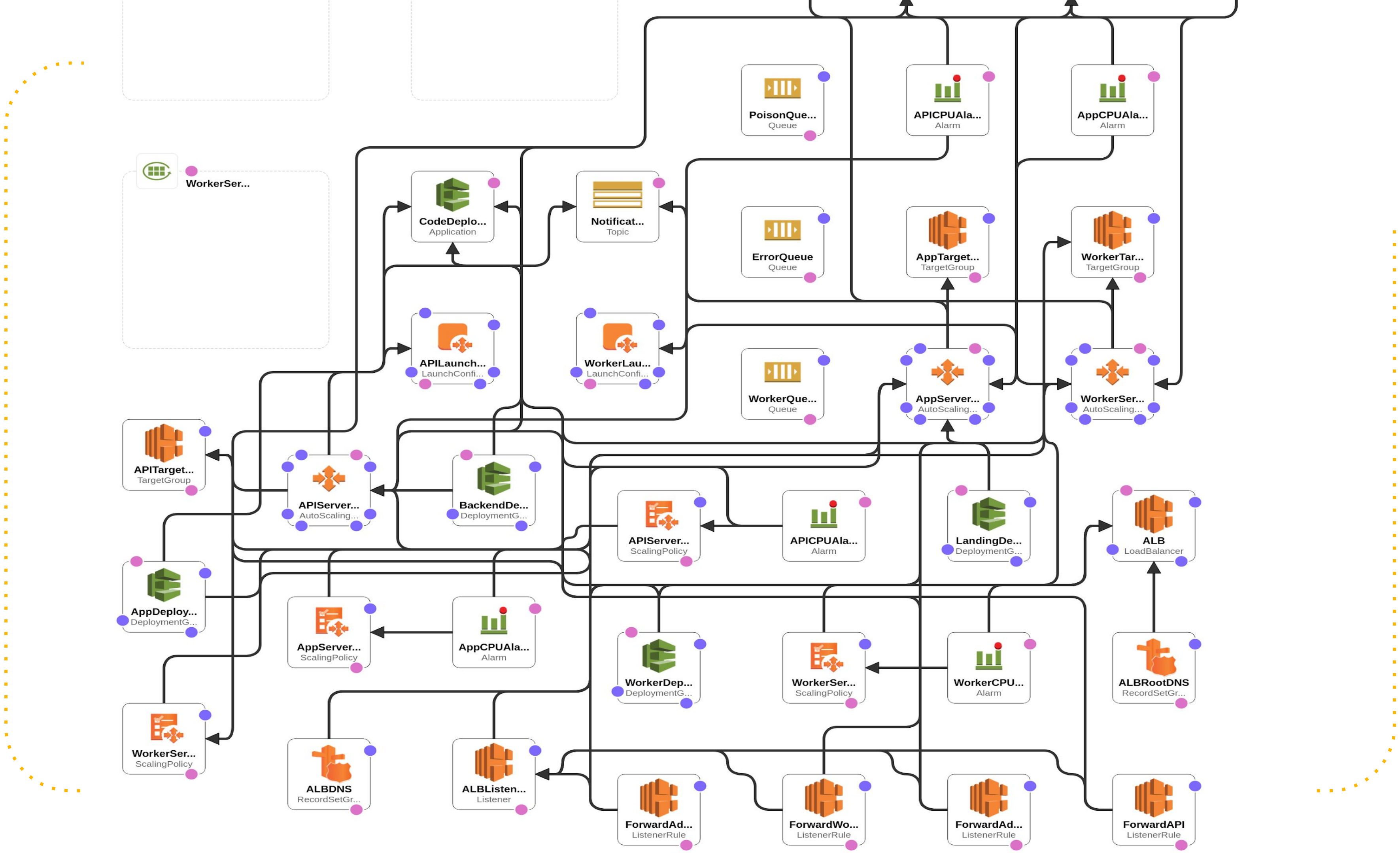
- We can have many different Dev instances
- Use in-between instance to get everything together

codesushi

How to - templates

```json
{
    "Resources": {
        "HelloBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "MyBucket"
            }
        }
    }
}
```

```yaml
Resources:
 HelloBucket:
   Type: AWS::S3::Bucket
   Properties:
     BucketName: MyBucket
```


WorkerSer...
PoisonQue...
Queue
APICPUAla...
Alarm
AppCPUAla...
Alarm
CodeDeplo...
Application
Notificat...
Topic
ErrorQueue
Queue
AppTarget...
TargetGroup
WorkerTar...
TargetGroup
APILaunch...
LaunchConfi...
WorkerLau...
LaunchConfi...
WorkerQue...
Queue
AppServer...
AutoScaling...
WorkerSer...
AutoScaling...
APITarget...
TargetGroup
APIServer...
AutoScaling...
BackendDe...
DeploymentG...
APIServer...
ScalingPolicy
APICPUAla...
Alarm
LandingDe...
DeploymentG...
ALB
LoadBalancer
AppDeploy...
DeploymentG...
AppServer...
ScalingPolicy
AppCPUAla...
Alarm
WorkerDep...
DeploymentG...
WorkerSer...
ScalingPolicy
WorkerCPU...
Alarm
ALBRootDNS
RecordSetGr...
WorkerSer...
ScalingPolicy
ALBDNS
RecordSetGr...
ALBListen...
Listener
ForwardAd...
ListenerRule
ForwardWo...
ListenerRule
ForwardAd...
ListenerRule
ForwardAPI
ListenerRule

Example Symfony stack

Initializing Instance

- CloudFormation::init
    - Running custom commands
    - Installing and configuring packages
    - Creating new files

- No need to customize AMI

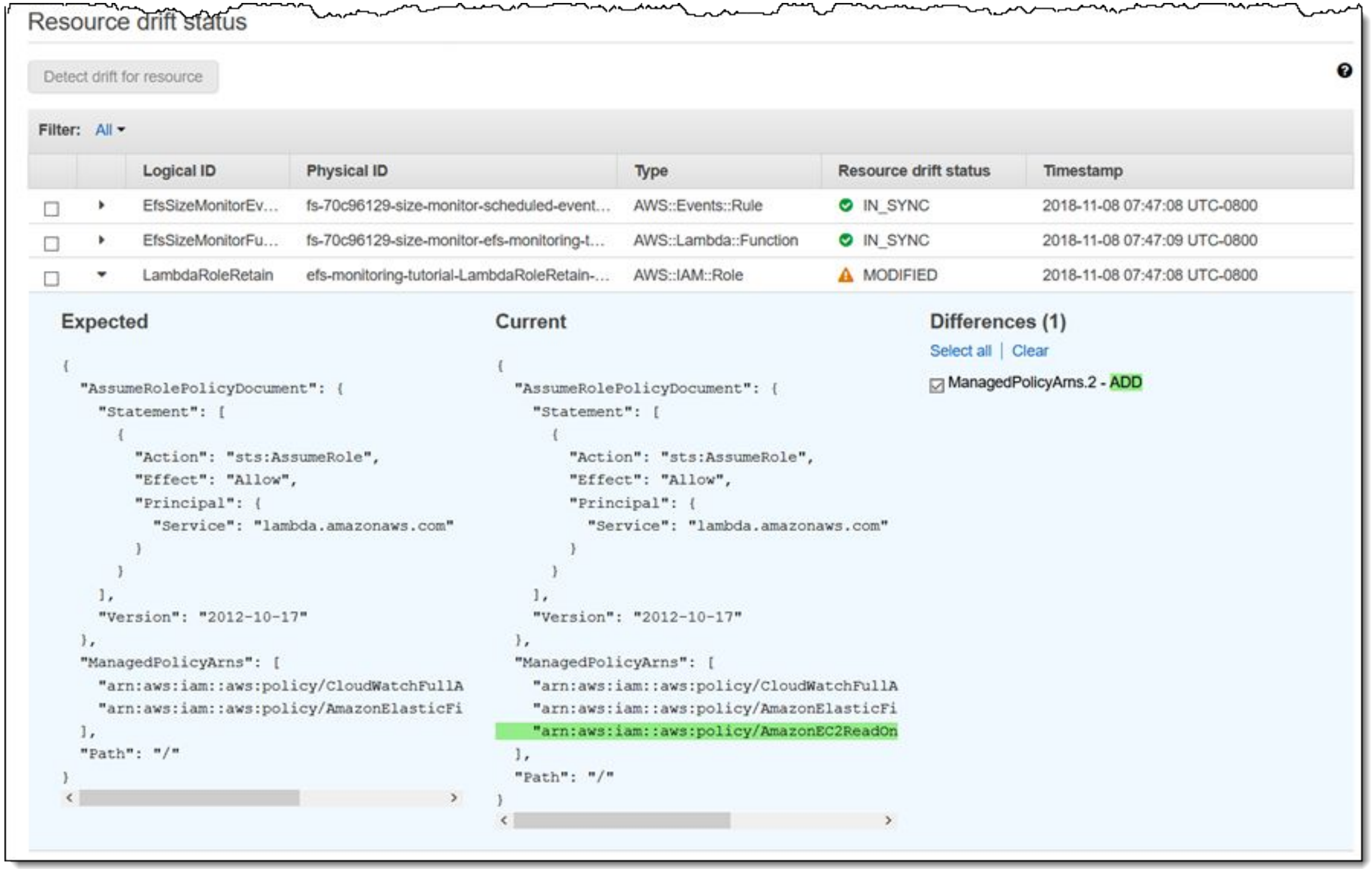
codesushi

Drift(ing apart)

In general we shouldn't update Resources directly.

Doing so risks introducing variability in the infrastructure, which will be lost during next proper CF update.

Hotfixes are fine - **BUT** - later they should be introduced by proper channels.



Resource drift status

Filter: All

		Logical ID	Physical ID	Type	Resource drift status	Timestamp
☐	▸	EfsSizeMonitorEv...	fs-70c96129-size-monitor-scheduled-event...	AWS::Events::Rule	✓ IN_SYNC	2018-11-08 07:47:08 UTC-0800
☐	▸	EfsSizeMonitorFu...	fs-70c96129-size-monitor-efs-monitoring-t...	AWS::Lambda::Function	✓ IN_SYNC	2018-11-08 07:47:09 UTC-0800
☐	▾	LambdaRoleRetain	efs-monitoring-tutorial-LambdaRoleRetain-...	AWS::IAM::Role	⚠ MODIFIED	2018-11-08 07:47:08 UTC-0800

Expected

```
{
  "AssumeRolePolicyDocument": {
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "lambda.amazonaws.com"
        }
      }
    ],
    "Version": "2012-10-17"
  },
  "ManagedPolicyArns": [
    "arn:aws:iam::aws:policy/CloudWatchFullA
    "arn:aws:iam::aws:policy/AmazonElasticFi
  ],
  "Path": "/"
}
```

Current

```
{
  "AssumeRolePolicyDocument": {
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "lambda.amazonaws.com"
        }
      }
    ],
    "Version": "2012-10-17"
  },
  "ManagedPolicyArns": [
    "arn:aws:iam::aws:policy/CloudWatchFullA
    "arn:aws:iam::aws:policy/AmazonElasticFi
    "arn:aws:iam::aws:policy/AmazonEC2ReadOn
  ],
  "Path": "/"
}
```

Differences (1)

Select all | Clear

☑ ManagedPolicyArns.2 - ADD

codesushi

Create Change Set

The safe way to commit changes to Stacks.

Will list required actions before doing them.

Needs our confirmation and can be dismissed.

We can always do the update directly, but why risking going ALL IN on production?

codesushi

Staying together

- Stack Policies
- DeletionPolicy - delete, retain, snapshot
- Termination protection - a must have for production

codesushi

How can I hold all these templates?

codesushi

Version Control

- Use repository to keep track of changes in the templates.
- Infrastructure as/is a code

  Git repo, AWS S3
- Track any useful partials - helping scripts, parameters, configs.

How to start

- Existing templates (*divide and conquer - copy and paste*)
- Designer
- Writing manually
  - Linters (cfn-lint)
  - Static analysers (cfn_nag)
  - Libraries (eg. troposphere)

Moving to CF

Analyze your current infrastructure

Single out the elements you need and what exactly composes them

Model them in CF template

codesushi

Any Questions?

codesushi

Thanks for listening

andrew.m@codesushi.co