

Notejam

Deployment Pipeline Architecture



Deployment Pipeline

DevOps Standard CI & CD

Summary

- Immutable Golden Image Pipeline
- Automatic deploys to master
- Manual promotion to master
- CloudWatch Telemetry
- Ansible & CloudFormation for orchestration
- Postgres RDS for database
- Using Rails sample app

Build w/Packer

- Packer is the industry standard
- Ansible playbook (pipeline/ configure-ami.yml) does the heavy lifting
- AMI includes logging and telemetry config
- AMI usable across all pipeline stages

```
2  variables: {
3
4  },
5  builders: [{
6    type: amazon-ebs,
7    region: ap-southeast-1,
8    source_ami_filter: {
9      filters: {
10        virtualization-type: hvm,
11        name: ubuntu/images/*ubuntu-xenial-16.04-amd64-server-*
12        root-device-type: ebs
13      },
14      owners: [099726109477],
15      most_recent: true
16    },
17    instance_type: c5.large,
18    ssh_username: ubuntu,
19    ami_name: toptal-notejam {{timestamp}}
20  },
21  provisioners: [{
22    type: file,
23    source: {{ user `code_archive` }},
24    destination: /tmp/code.tar.gz
25  }, {
26    type: shell,
27    inline: sudo apt-get update && sudo apt-get install -y soft
28  }, {
29    type: ansible-local,
30    playbook_file: pipeline/configure-ami.yml,
31    playbook_dir: pipeline,
32    extra_arguments: [
33      --extra-vars app_archive=/tmp/code.tar.gz
34    ]
35  },
36  post-processors: [{
37    type: manifest,
38    output: tmp/manifest.json,
39    strip_path: true
40  }
41  }
```

Deploy w/CloudFormation

- Infrastructure as Code from t-zero
- Postgres RDS instance
- 2AZ setup with ALB
- Autoscaling Triggers
- Reused for all deployment stages
- Running app tested in deployment pipeline

```
Port: 80
Protocol: HTTP
DBSecurityGroup:
  Type: "AWS::EC2::SecurityGroup"
  Properties:
    GroupDescription: "DB security group"
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: TCP
        FromPort: !Ref DBPort
        ToPort: !Ref DBPort
        SourceSecurityGroupId: !GetAtt ServerSecurityGroup.GroupId
DBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: "Subnets available for the RDS DB Instance"
    SubnetIds:
      - !Ref SubnetA
      - !Ref SubnetB
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: postgres
    EngineVersion: "9.6.9"
    DBInstanceClass: db.m4.large
    AllocatedStorage: 100
    MasterUsername: !Ref AppName
    MasterUserPassword: !Ref DBPassword
    DBSubnetGroupName: !Ref DBSubnetGroup
    Port: !Ref DBPort
    VPCSecurityGroups:
      - !Ref DBSecurityGroup
LaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateName: !Sub "${AppName}-${Stage}"
    LaunchTemplateData:
      ImageId: !Ref AMI
      InstanceType: !Ref InstanceType
      KeyName: !Ref KeyName
      SecurityGroupIds:
        - !GetAtt ServerSecurityGroup.GroupId
      IamInstanceProfile:
        Name: !Ref InstanceProfile
      TagSpecifications:
        - ResourceType: instance
          Tags:
            - Key: Name
              Value: !Sub "${AppName}-${Stage}-app"
    UserData:
```

[Sign In](#) [Sign Up](#)

notejam: Sign In

Please sign in

Email

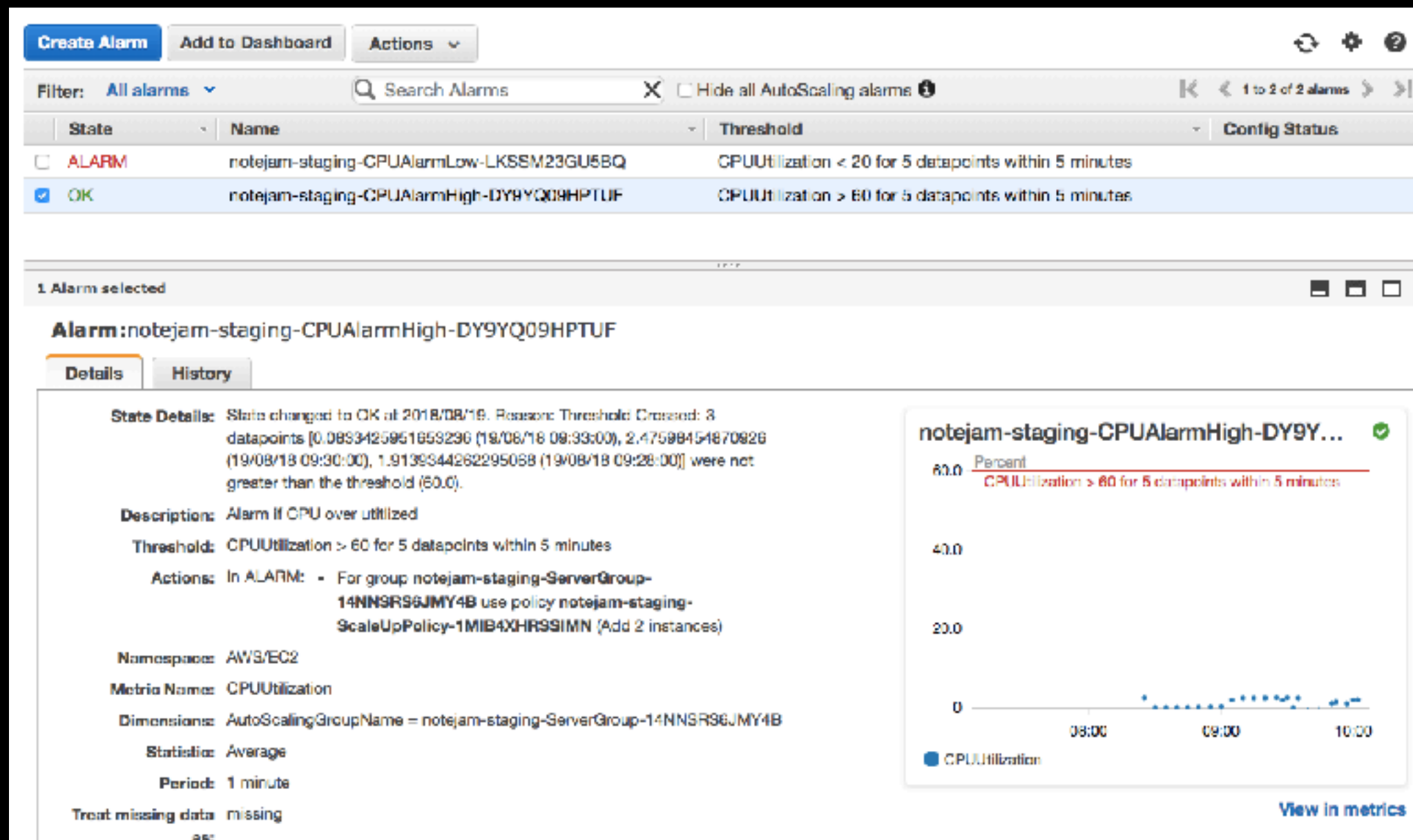
Password

or [Sign Up](#)

[Forgot password?](#)

App in Production

Each deploy tests infra with GET /ping



AutoScaling Alarms

Commit to CloudFormation template and deploy

| | Time (UTC-0800) | Message |
|---|-----------------|--|
| | 2018-08-19 | |
| ▶ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |
| ▶ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |
| ▶ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |
| ▶ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |
| ▼ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |
| | | { |
| | | "instanceId": "i-03fae5b5d3b424c69", |
| | | "pid": 1337, |
| | | "uid": 1001, |
| | | "gid": 1002, |
| | | "cmdName": "ruby2.3", |
| | | "exe": "/usr/bin/ruby2.3", |
| | | "cmdLine": "puma 3.12.0 (tcp://0.0.0.0:8080) [app] ", |
| | | "systemdUnit": "app.service", |
| | | "bootId": "6a7e40c668be4be895d5316b4814cad1", |
| | | "hostname": "ip-192-168-1-223", |
| | | "transport": "stdout", |
| | | "priority": 6, |
| | | "message": "Started GET \"/ping\" for 192.168.1.116 at 2018-08-19 10:06:15 +0000", |
| | | "seq": 979, |
| | | "syslogFacility": 3, |
| | | "syslogIdent": "app" |
| | | } |
| ▶ | 10:06:15 | { "instanceId": "i-03fae5b5d3b424c69", "pid": 1337, "uid": 1001, "gid": 1002, "cmdName": |

Remote Log Access

Stream logs by CW utils

Next Steps

- Review Technology Choices
- Lock down SSH access
- Speed up deployment pipeline by using a base AMI
- Evaluate Spinnaker or alternate deployment options (e.g. containers)
- Create bastion for application console access
- Configure cost allocation tags
- Create per-topic branch environments