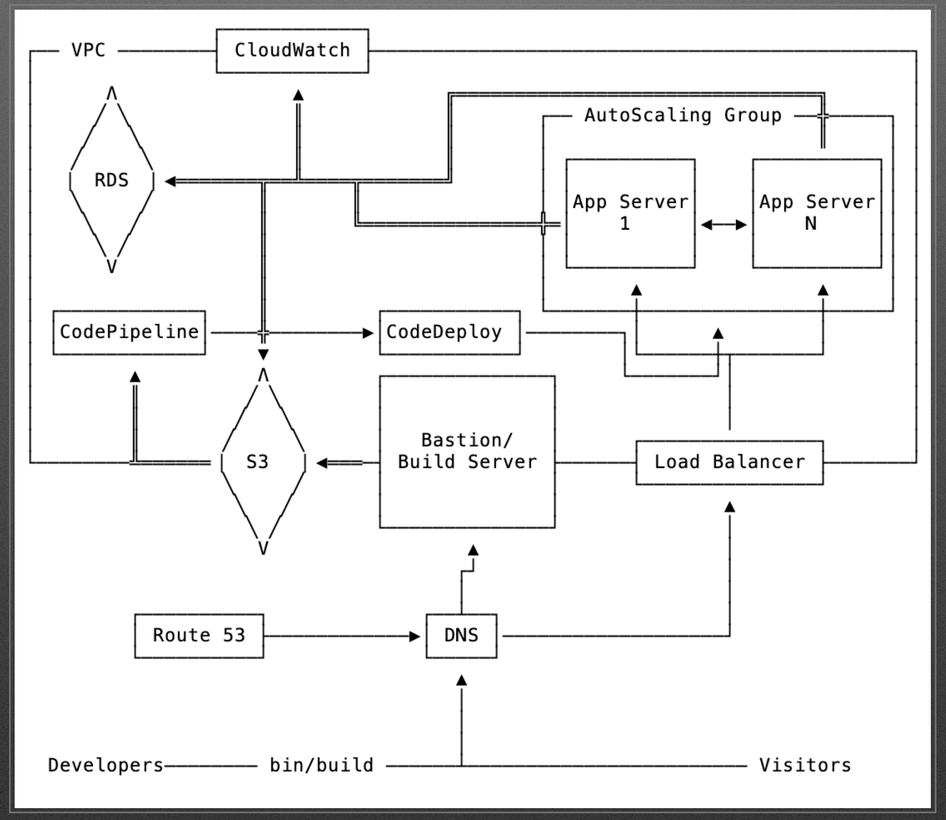
The Details

Elixir in the Jungle

Architecture



Our Plan

- Slack Integration–notifications: Simple
- Slack Integration–CodeDeploy status: More Complex
- Logging with CloudWatch
- Q&A

"How do you notify Slack?"

Setup a Slack Webhook

https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX

```
  - path: /usr/bin/notify_slack
    owner: root:root
    permissions: '0755'
    content: |
      #!/bin/bash
      curl -s -X POST -H 'Content-type: application/json' --data "{\"text\":\"$1\"}" $SLACK_WEBHOOK
```

Notify Slack

CodeDeploy Status to Slack

Services: SNS and Lambda

- SNS is a pub/sub messaging service
  - This is valuable for all manner of service communication
  - CodeDeploy can publish to SNS topics as deploys progress
- Lambda is a service for running some code in response to events without provisioning more servers

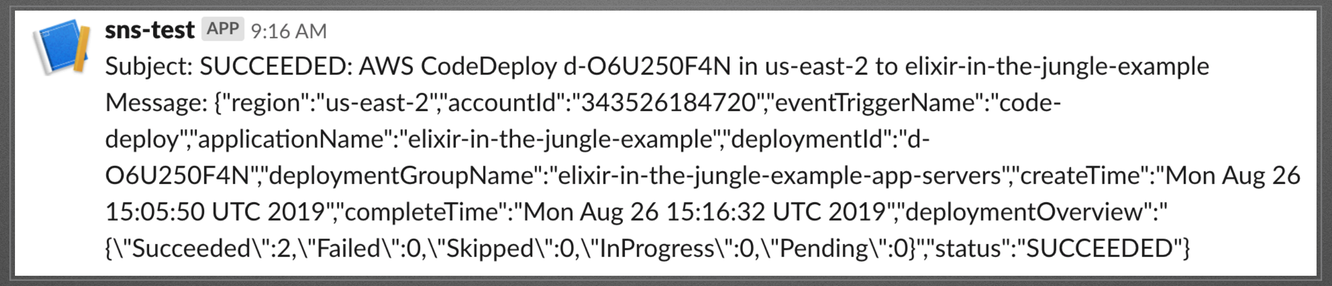
```
var https = require('https');

exports.handler = async (event, context) => {

    const subject = event.Records[0].Sns.Subject;
    const message = event.Records[0].Sns.Message;

    const postData = JSON.stringify({ text:"Subject: " + subject + "\nMessage: " + message});

    return new Promise((resolve, reject) => {
        const options = {
          method: 'POST',
          hostname: 'hooks.slack.com',
          port: 443,
          path: process.env.webhook,
          headers: {
            'Content-Type': 'application/json'
          }
        };

        const req = https.request(options, (res) => { resolve('Success'); });
        req.on('error', (e) => { reject(e.message); });

        // send the request
        req.write(postData);
        req.end();
    });
};
```

SNS to Slack

Via Lambda

```
resource "aws_sns_topic_subscription" "code_deploy_updates" {
  topic_arn = "${aws_sns_topic.app_servers.arn}"
  protocol  = "lambda"
  endpoint  = "${aws_lambda_function.slack_lambda.arn}"
}

resource "aws_lambda_function" "slack_lambda" {
  filename      = "sns_to_slack.zip"
  function_name = "sns_to_slack"
  role          = "${aws_iam_role.iam_for_lambda.arn}"
  handler       = "index.handler"

  source_code_hash = "${filebase64sha256("sns_to_slack.zip")}"

  runtime = "nodejs10.x"

  environment {
    variables = {
      webhook = "/services/TEW1PRACD/BMSERDBLN/LPPzNlwdQZ0H8CtxlacQbZYw"
    }
  }
}
```

Terraform Lambda



A Slack Notification!

Log Aggregation

We Collect Logs in CloudWatch

- We use systemd to manage our application on instances
- CloudWatch requires an add-on Go program to get along with systemd
  - https://github.com/advantageous/systemd-cloud-watch
- We precompiled it for our instance OS, stuck the binary in S3, and we download and start it as the instance comes up
- You don't have to use this log setup (Papertrail is an alternative)

Service: CloudWatch

- CloudWatch is a monitoring service
- It collects logs, metrics, and events
- It supports setting alarms or taking automated actions when configured criteria are met

```
#!/usr/bin/env bash

set -e

# This should really be built and hosted in your own internal storage
# to prevent certain types of attacks and instability
sudo wget -O /usr/local/bin/systemd-cloudwatch-logs \
     https://github.com/advantageous/systemd-cloud-watch/releases/download/v0.2.1/systemd-cloud-watch_linux

sudo chmod +x /usr/local/bin/systemd-cloudwatch-logs

sudo systemctl enable cloudwatch_logs
sudo systemctl start cloudwatch_logs
```

Installing the CloudWatch Helper

Configured by cloud-init

Your Turn to Drive!

Tell us which pieces you would like to explore more…