Infrastructure Automation: Real Examples

BIOTEAM
Enabling Science

Karl Gutwin
Sr Scientific Consultant, BioTeam

We live in an automated world

infrastructure automation is robotics for IT

BIOTEAM
Enabling Science

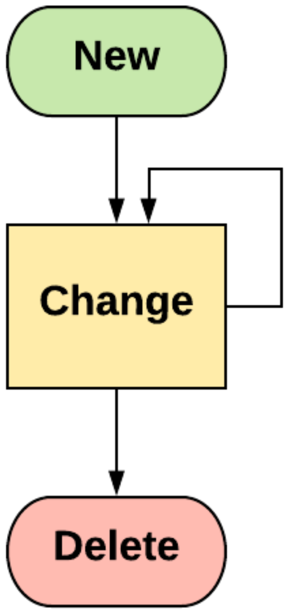
Infrastructure as Code

Know the intent and implementation of your system

BIOTEAM
Enabling Science

Automation is more than provisioning

- Managing changes
- Cleanup and deletion



BIOTEAM
Enabling Science

Why not automation?

- Additional point of failure
- Too clever for its own good
- Code as a liability
- Sometimes additional costs

These are all risks we currently manage within IT

BIOTEAM
Enabling Science

Many tools to choose from

including

- Ansible
- Docker
- Lambda
- CloudFormation

BIOTEAM
Enabling Science

Examples

↳ github.com/bioteam/infrastructure-automation

Ansible

Configure your server

webserver.yml

```yaml
---
- hosts: localhost
  tasks:
    - name: install nginx
      package: name=nginx state=present

    - name: set to run on boot
      service: name=nginx enabled=yes state=started
```

database.yml

```yaml
---
- hosts: localhost
  tasks:
    - name: install postgresql
      package: name=postgresql state=present

    - name: update config file
      lineinfile:
        path: /etc/postgresql/9.6/main/postgresql.conf
        regexp: '^synchronous_commit ='
        line: 'synchronous_commit = {{ pg_sync|default("on") }}'
      register: postgres_config

    - name: set to run on boot
      service: name=postgresql enabled=yes state=started

    - name: restart service
      service: name=postgresql state=restarted
      when: postgres_config.changed
```

```
ansible-playbook database.yml
```

Get Started

↳ `pip install ansible`

↳ github.com/ansible/ansible-examples

↳ docs.ansible.com/ansible/latest/user_guide/playbooks_intro.htm

Docker

Applications in Containers

Dockerfile

```
FROM node

RUN git clone https://github.com/hakimel/reveal.js.git && \
    cd reveal.js && npm install

EXPOSE 8000
WORKDIR /reveal.js
CMD ["npm", "start"]
```

```
docker build -t my-image .
```

docker-compose.yml

```yaml
---
version: '3'
services:
  mediawiki:
    image: mediawiki
    restart: always
    ports: ['8080:80']
    links: ['database']
    volumes:
      - /var/www/html/images
#     - ./LocalSettings.php:/var/www/html/LocalSettings.php
  database:
    image: mariadb
    restart: always
    environment:
      MYSQL_DATABASE: my_wiki
      MYSQL_USER: wikiuser
      MYSQL_PASSWORD: example
      MYSQL_RANDOM_ROOT_PASSWORD: 'yes'
```

```
docker-compose up
```

Get Started

↳ www.docker.com/products/docker-desktop

↳ docs.docker.com/compose/gettingstarted/

Lambda

Event-driven automation

autotag.py

```python
import boto3
ec2 = boto3.client('ec2')

def lambda_handler(event, context):
    principal_type = event['detail']['userIdentity']['type']
    if principal_type == 'IAMUser':
        username = event['detail']['userIdentity']['userName']
    else:
        username = event['detail']['userIdentity']['principalId']

    response = event['detail']['responseElements']
    instance_ids = [i['instanceId']
                    for i in response['instancesSet']['items']]

    ec2.create_tags(Resources=instance_ids,
                    Tags=[{"Key": "Owner", "Value": username}])
```

```
demo-autotag.sh
```

Other ideas

- Manage DNS entries
- Integrate with a CMDB
- Automate application rollout
- Automatic health checks for critical applications
- Data pipelines and ETL

Get Started

↳ docs.aws.amazon.com/AmazonCloudWatch/latest/events/CloudWatch-Events-Tutorials.html

CloudFormation

AWS infrastructure templates

bucket.yml

```yaml
---
AWSTemplateFormatVersion: "2010-09-09"
Parameters:
  BucketName:
    Type: String
    Default: my-awesome-bucket

Resources:
  StaticBucket:
    Type: "AWS::S3::Bucket"
    Properties:
      BucketName: !Ref BucketName
```

whatismyip.yml

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Transform: "AWS::Serverless-2016-10-31"
Resources:
  WhatIsMyIP:
    Type: "AWS::Serverless::Function"
    Properties:
      Handler: index.handler
      Runtime: python3.7
      InlineCode: |
        import json
        def handler(event, context):
          body = {'ip': event['requestContext']['identity']['sourceIp']}
          return {'statusCode': '200', 'body': json.dumps(body)}
      Events:
        Web:
          Type: Api
          Properties:
            Path: /
            Method: get
```

```
aws cloudformation create-stack

--stack-name whatismyip --template-body file://whatismyip.yml
--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_IAM
```

Get Started

↳ docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/gettingstarted.templatebasics.html

↳ aws.amazon.com/serverless/sam

Thanks!

BIOTEAM
Enabling Science

bioteam.net