MASTER & MINIONS
OR THE DREAM OF PYTHON AUTOMATION

Calvin Hendryx-Parker
CTO, Co-Founder
Six Feet Up
CERTAIN THINGS SHOULD BE HAND CRAFTED.
BUT NOT YOUR INFRASTRUCTURE
BEAUTIFUL UNIQUE SNOWFLAKES ARE NOT REPRODUCIBLE.
CATTLE
RULES OF DEVOPS CLUB

» The first rule of DevOps Club is: You do not log into servers
» The second rule of DevOps Club is: You do not log into servers
» Third rule of DevOps Club: If your deployment fails, rollback
» Fourth rule: All artifacts will be stored in source control
» Fifth rule: Only one deployment at a time
» Sixth rule: No one offs, No special cases
» Seventh rule: Deployments will go on as long as they have to
» And the eighth and final rule: If this is your first night at DevOps Club, you have to push to prod.
To enter the DevOps world, you need to know what tools are available to you.

Python has many great tools available to use such as SaltStack and the AWS Boto3 library.
SINGLE SERVER MONOLITH

Web/App/Database
- Nginx
- Varnish
- HAPerxy
- Plone Instances x 4
- ZEO Storage Server
FROM THE CLOSET TO THE CLOUD
Cloud Optimized

CloudFront

ELB

AZ 1
HAPerxy (rewrite urls)
Plone App Server

EFS (blobstorage)

AZ 2
HAPerxy (rewrite urls)
Plone App Server

RDS Multi-AZ Master

S3 Backup
Read This and Learn How to Button Your Collar in a Hurry

AUTOMATION

SIX FEET UP — MASTER & MINIONS, OR THE DREAM OF PYTHON AUTOMATION

STAND NEAR POOL TABLE (A) — BALL (C) FALLS THROUGH POCKET (B) STARTING SEE-SAW (D) AND BOUNCING MEASLES-GERM (E) INTO THE AIR — GERM HITS DOLL (I) WHICH IMMEDIATELY CATCHES MEASLES AND DEVELOPS A HIGH FEVER — FEVER HEATS COFFEE POT (G) AND COFFEE BOILS OVER — DRIPPING THROUGH FUNNEL (I) INTO CAT'S MOUTH GIVING CAT (J) INSOMNIA — CAT CLIMBS TREE (K) FOR DIVERSION AND KNOCKS OFF BRICK (L) — BRICK HITS CHILD (M) IN HEAD — CHILD PRESSES BUTTON (N) THINKING IT WILL ATOMISE — BUTTON SETS UP CURRENT IN COIL (O) WHICH CHARGES MAGNET (P) HOLDING COLLAR SECURELY IN POSITION.
ENTER SALTSTACK AND BOTO3
# VPC

c = boto.vpc.connect_to_region(region)
vpc = c.create_vpc('172.20.0.0/16')
vpc.add_tag('Name', 'control-vpc')
c.modify_vpc_attribute(vpc.id, enable_dns_hostnames=True)
$ $(lpass-env export 'AWS SFU Calvin')

$ env | grep AWS
AWS_ACCESS_KEY_ID=DEADBEEFCAFE
AWS_SECRET_ACCESS_KEY=123jasdfads0of9akayo5peey0cow
AWS_DEFAULT_REGION=us-east-1
```python
# EC2
data_path = os.path.join(os.path.dirname(__file__), 'bootstrap-master.sh')
with open(data_path) as script:
    bootstrap = script.read()

master = ec2.run_instances(
    'ami-55ef662f',
    subnet_id=subnet.id,
    security_group_ids=[group.id],
    key_name=ssh_key,
    instance_type='t2.micro',
    user_data=bootstrap
)

master.instances[0].add_tag('Name', 'Salt Master')
```
WHAT IS SALTSTACK?
WHAT SETS SALT APART?

» Remote Execution
» Event-Driven Orchestration
» Agent or Agent-less Operation
» Cloud Provisioning
» Speed and Scalability
NOW WE ORCHESTRATE

$ salt-run state.orchestrate orch.deploy-environment pillarenv=prod

And build a test environment

$ salt-run state.orchestrate orch.deploy-environment pillarenv=test
MASTER AND MINIONS

SIX FEET UP — MASTER & MINIONS, OR THE DREAM OF PYTHON AUTOMATION
ORCHESTRATING NEW CODE RELEASES
ZERO DOWNTIME RELEASES

{% for app_server in app_servers %}
...
# Stop Varnish, indirectly removing this app server from the NetScaler
stop-{{ app_server }}-varnish:
salt.function:
  - name: service.stop
  - arg:
    - {{ varnish.service }}
  - tgt: {{ app_server }}
...

PREPARE FOR THE RELEASE

# Stop Instances
stop-{{ app_server }}-instances:
salt.function:
  - name: supervisord.stop
  - arg:
    - all
  - tgt: {{ app_server }}
OPTIONAL AD-HOC CODE RELEASES

...

# Check out a specific revision
{% set branch = salt['pillar.get']('plone:branch', 'dev') %}
# to specify on the command-line: pillar='"plone": {"branch": "f5d9859"}'}
checkout-code-{{ app_server }}:
salt.function:
  - tgt: {{ app_server }}
  - name: git.checkout
  - kwarg:
    user: webuser
    rev: {{ branch }}
  - arg:
    - {{ buildout_dir }}
...

SIX FEET UP — MASTER & MINIONS, OR THE DREAM OF PYTHON AUTOMATION
HANDLE RELEASE TASKS

# Run the buildout on the app server, but only run the generic setup on one server
run-buildout-{{ app_server }}:
  salt.function:
    - tgt: {{ app_server }}
    - name: cmd.run
    - kwarg:
      - cwd: {{ buildout_dir }}
        runas: webuser
      - {{ buildout_dir }}/env/bin/buildout -N
# Install Gulp bits
install-gulp-{{ app_server }}:
  salt.function:
    - name: npm.install
    - tgt: {{ app_server }}
    - kwarg:
      runas: webuser
      dir: {{ theme_dir }}

# Run the Gulp Build
run-gulp-{{ app_server }}:
  salt.function:
    - tgt: {{ app_server }}
    - name: cmd.run
    - arg:
      gulp build
    - kwarg:
      runas: webuser
      cwd: {{ theme_dir }}
NOT JUST FOR RELEASES

» Migrating Production Data back to Testing
» Scheduled tasks such as Database Backups
policies:
  - name: offhours-stop-ec2
    mode:
      type: periodic
      schedule: "rate(1 hour)"
      role: arn:aws:iam::243886768005:role/cloud_custodian
    resource: ec2
    filters:
      - type: offhour
        default_tz: America/Indiana/Indianapolis
        offhour: 16
    actions:
      - stop
GRABBING PRODUCTION DATA

# Find the dev master db for use later
{% set ns = namespace(dev_master=None) %}
{% for dev_master in dev_db_servers
    if salt.saltutil.cmd(dev_master, 'file.search',
        arg=['/var/run/keepalive.state', 'MASTER']).get(dev_master).get('ret') %
    {% if loop.first %
        {% set ns.dev_master = dev_master %}
    {% endif %}
    {% endif %}
{% endfor %}

# Transfer Prod database
{% for prod_slave in prod_db_servers
    if salt.saltutil.cmd(prod_slave, 'file.search',
        arg=['/var/run/keepalive.state', 'BACKUP']).get(prod_slave).get('ret') %
    {% if loop.last %
        sync-prod-database-to-dev:
        salt.function:
            - name: cmd.run
            - arg:
              -
                rsync --delete --password-file=/etc/rsyncd.password.dump
                /u02/prodplonedb_archives/ dump@{{ ns.dev_master }}::postgres_dumps
            - tgt: {{ prod_slave }}
    {% endif %}
{% endfor %}
SOUNDS TOO EASY

The road was bumpy for sure.

Satisfying the rules for no special cases was tricky.
MINDFULNESS

“There should be one, and preferably only one, obvious way to do it. Although that way may not be obvious at first unless you're Dutch.”

The Zen of Python
THE JOURNEY OF 3 OPERATING SYSTEMS

- 2a67758 Editing requirements to run properly on Amazon Linux
- 472d844 Refactoring to run CentOS 7 machines
- dd67b7a Refactoring for FreeBSD

What happened here?
OTHER RECOMMENDATIONS

» Implement a CI strategy to test your infrastructure with tools like Kitchen-Salt

» Create or use tools to help you trace requests in the cloud: https://github.com/sixfeetup/aws-log-tools
THANKS!
CALVIN@SIXFEETUP.COM
@CALVINHP