Programming the Cloud

Data Center Migration

Opening Question 2

Development vs Operations

Opening Question 1

Should we use cloud technologies? **Why?**

Goal

Smooth and Calm Flow of Value into Production

Goal

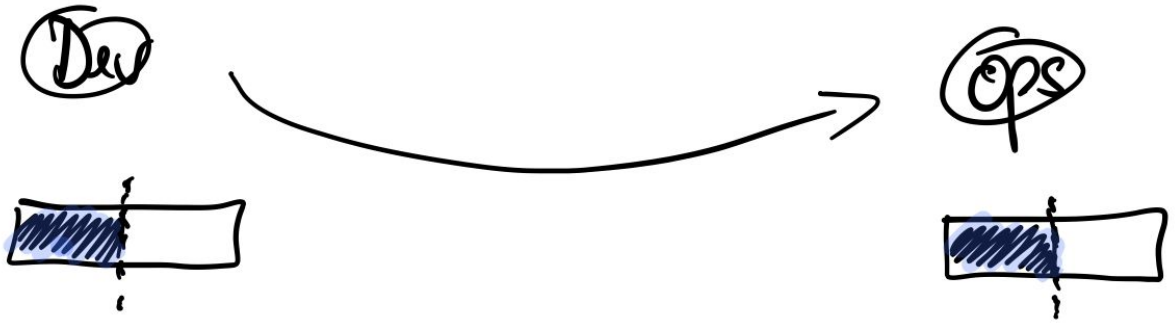
Sleep Calm at Night

Obstacles

- Chronic Conflict: Development vs Operations
- Left Behind: Everybody is transitioning to the cloud. We must as well!
- Accelerating Technology: What is here to stay? What will come next?
- Best Practices, Services: Lambda, S3, EC2, HostedZone, EKS, ECR, ...
- Best Practices, Scripting: AWS CLI, Python, Terraform, CloudFormation, …
- Learning Curve: It takes time to master new stuff.
- Over-Engineering: Getting distracted by too much cool stuff :-)

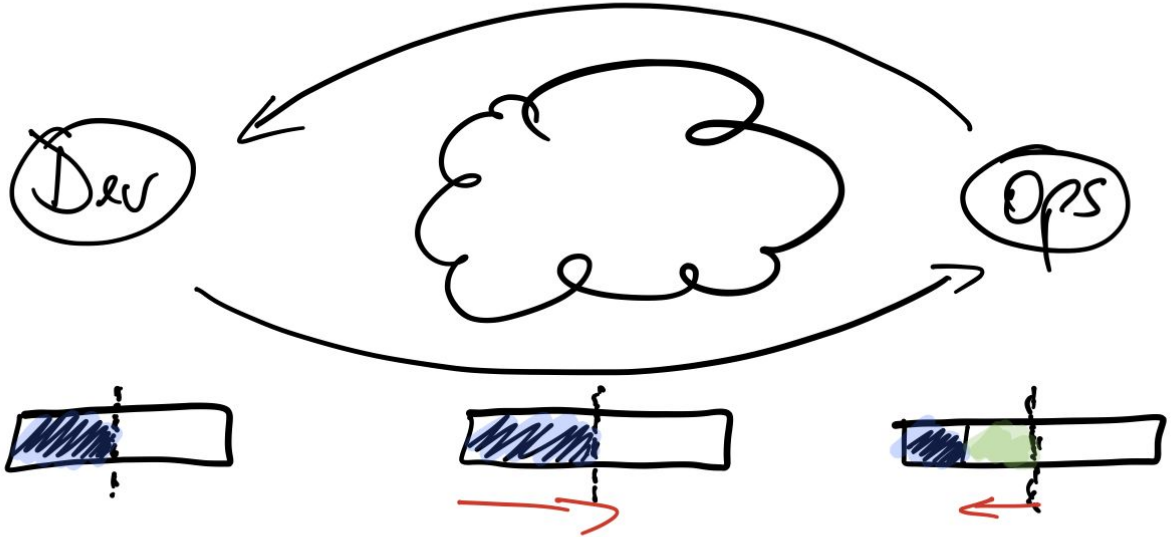
Stay Focused

Solution 1



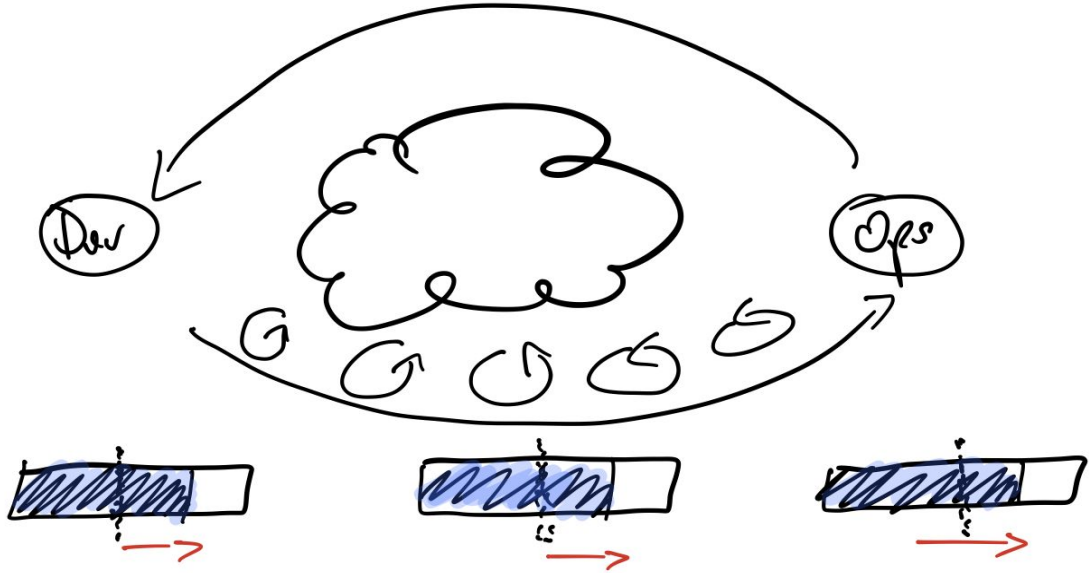
- Hot Patching: No test/staging environments => Danger Zone!
- Details, Complexity, Mistakes, Know-How Silos => Stress, Avoidance
- Ego-Fights: Devs vs Ops
- Long Release Cycles
+ Pragmatic: Get It Done Mentality / Quick and Dirty

Solution 1

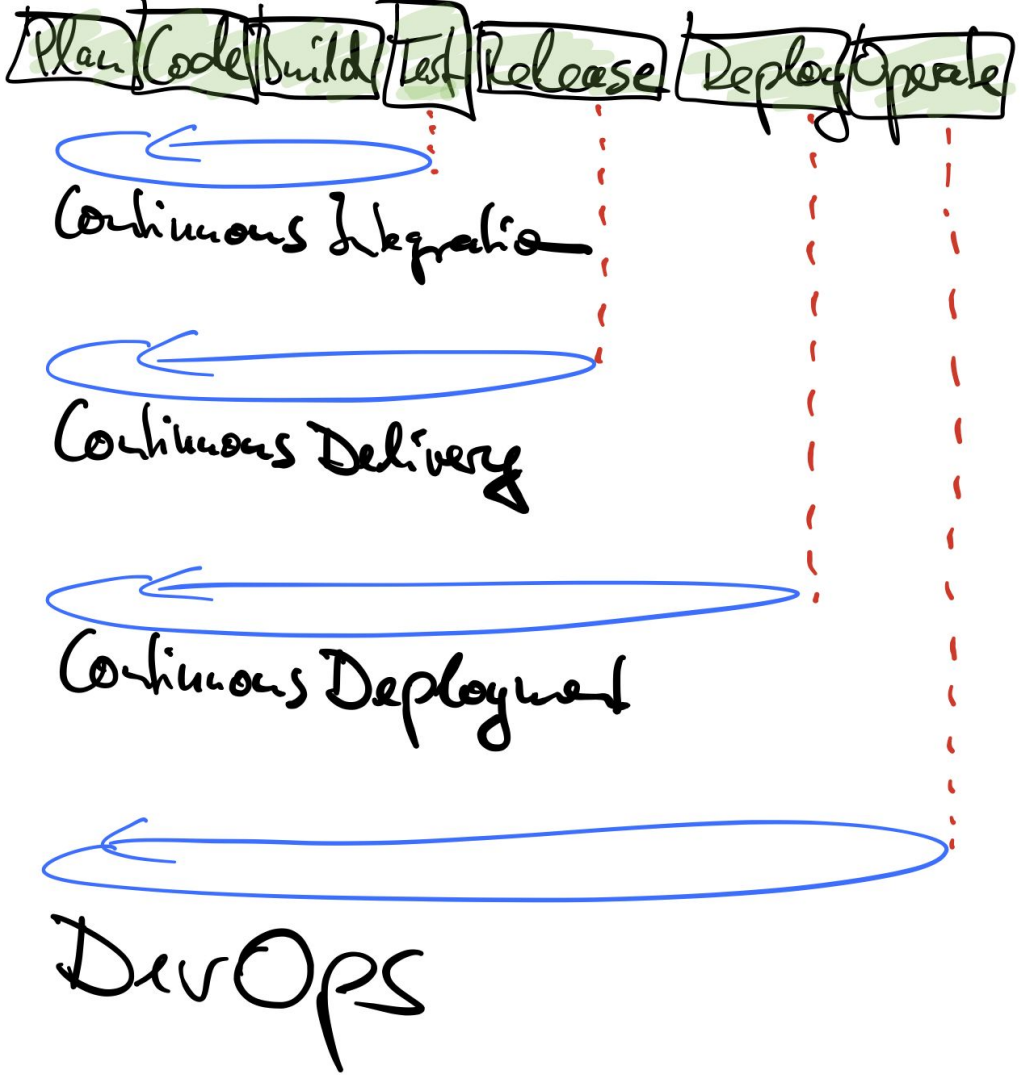


+ Cooperation: Dev + Ops
+ Offload Heavy the Lifting into the Cloud => Uptime
+ $$$: Cost Distribution

Solution 1



+ **Velocity**
+ **Documentation**
+ New Features/Improvements
+ Cooperation
+ Automation
+ Uptime
+ High performing individuals

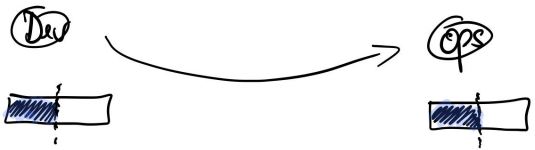


Code Pipeline

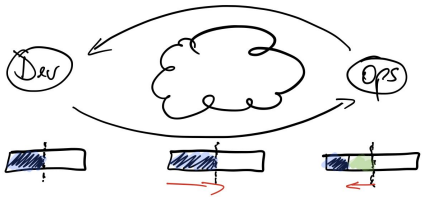
Let’s Create a Plan

Where do you want to be?

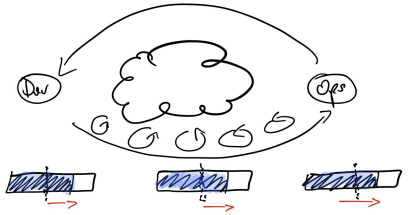
1



2



3



Appendix A

- AWS CLI
- AWS SDK for Python
- Terraform
- CloudFormation
- Pulumi

AWS CLI

```
/scripts> aws ec2 run-instances \
          --image-id ami-28422647 \
          --block-device-mappings DeviceName=/dev/sda1,Ebs='{VolumeSize=50}' \
          --instance-type t2.medium \
          --key-name rancher-user \
          --security-group-ids sg-0366cf0d6bc848c80
```

```
florian@Florians-MBP ~/g/s/g/h/d/scripts> aws ec2 describe-instances
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "ec2-18-194-185-118.eu-central-1.compute.amazonaws.com",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2018-09-10T17:41:22.000Z",
                    "PublicIpAddress": "18.194.185.118",
                    "PrivateIpAddress": "172.31.33.254",
                    "ProductCodes": [],
                    "VpcId": "vpc-0224b009776c3ced2",
                    "StateTransitionReason": "",
                    "InstanceId": "i-0f696c69626448f60",
```

```
florian@Florians-MBP ~/g/s/g/h/d/scripts>
aws ec2 describe-instances \
| jq -r  ".Reservations[] | .Instances[] | .InstanceId"


i-0f696c69626448f60
i-0c5ebde78de0a4e87
florian@Florians-MBP ~/g/s/g/h/d/scripts>
```

```
florian@Florians-MBP ~/g/s/g/h/d/scripts> aws ec2 terminate-instances \
                                          --instance-ids i-0f696c69626448f60 i-0c5ebde78de0a4e87
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-0f696c69626448f60",
            "CurrentState": {
                "Code": 48,
                "Name": "terminated"
            },
            "PreviousState": {
                "Code": 48,
                "Name": "terminated"
            }
        },
```

AWS SDK for Python

```
#
# main
#

# build-server
instanceId = create_instance('t2.medium')
serverIp = get_public_ip(instanceId)
add_server_name(instanceId, 'build-server-x')
wait_for_ssh_available(serverIp)
wait_for_docker_is_alive(serverIp)
install_rancher_server(serverIp)
registrationToken = get_registration_token(serverIp)
rancher_access_key, rancher_secret_key = generate_api_key(serverIp)
enable_rancher_access_control(serverIp, RANCHER_USERNAME, RANCHER_PASSWORD)
install_rancher_host(serverIp, serverIp, "8080", registrationToken)
if AWS_HOSTED_ZONE_ID:
    update_dns_record(serverIp, AWS_DNS_NAME)                # 'hvt.zone.'
    update_dns_record(serverIp, '\\052.' + AWS_DNS_NAME)     # '\\052.hvt.zone.'
```

```
184 def wait_for_docker_is_alive(publicIpAddress):
185     print_function_name()
186     print('connecting to:' + publicIpAddress + ' and checking if docker is alive')
187 
188     try:
189         child = pexpect.spawn('ssh -i ' + AWS_SSH_KEY_PATH + ' rancher@' + publicIpAddress, encoding='utf-8')
190         child.logfile = sys.stdout
191         child.expect('rancher@ip-')
192 
193         while True:
194             child.sendline('docker info')
195             i = child.expect(['Live Restore Enabled', 'Cannot connect to the Docker daemon at unix'])
196             if i == 0:
197                 print("docker is running")
198                 break
199             if i == 1:
200                 print("\ndocker not fully started yet, retrying")
201                 time.sleep(5)
202                 continue
203         child.sendline("exit")
204         child.expect(pexpect.EOF)
205         child.close()
206     except Exception as e:
207         print('Exception: server not available')
208         print(e)
```

```python
def create_instance(instanceType):
    print_function_name()

    ec2 = boto3.resource('ec2',
                         region_name=AWS_REGION_NAME,
                         aws_access_key_id=AWS_ACCESS_KEY_ID,
                         aws_secret_access_key=AWS_SECRET_ACCESS_KEY)

    instances = ec2.create_instances(ImageId=AWS_IMAGE_ID,
                                     BlockDeviceMappings=[{"DeviceName": "/dev/sda1", "Ebs": {"VolumeSize": 50}}],
                                     InstanceType=instanceType,
                                     KeyName=AWS_SSH_KEY_NAME,
                                     SecurityGroupIds=[AWS_SECURITY_GROUP_ID],
                                     MinCount=1,
                                     MaxCount=1)
    print(instances)
    return instances[0].id
```

Terraform

```
resource "aws_instance" "rancher_host_instance" {
  ami = "${lookup(var.rancheros_amis, var.aws_region)}"
  instance_type = "${var.aws_instance_type}"
  tags = { Name = "rancher-host-${count.index}}" }
  security_groups = ["${var.aws_security_group}"]
  user_data = "${data.template_file.install_rancher_host.rendered}"
  key_name = "${var.aws_ssh_key_name}"
  root_block_device = {
    volume_size = "50"
    delete_on_termination = true
  }
  count = "${var.number_of_hosts}"
}
```

```
data "aws_route53_zone" "selected" {
  name = "${var.aws_hosted_zone}"
}

resource "aws_route53_record" "basic_hvt_zone" {
  zone_id = "${data.aws_route53_zone.selected.zone_id}"
  name    = "${data.aws_route53_zone.selected.name}"
  type    = "A"
  ttl     = "300"
  records = ["${aws_instance.rancher_server_instance.public_ip}"]
}
```

```
aws_route53_record.basic_hvt_zone: Creation complete after 1m0s (ID: ZQ6N4ERG12VKR_hvt.zone._A)
aws_route53_record.prefix_hvt_zone: Still creating... (1m0s elapsed)
aws_route53_record.prefix_hvt_zone: Creation complete after 1m2s (ID: ZQ6N4ERG12VKR_*.hvt.zone._A)

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:

rancher_access_key = 9731AEC93A633BEF08AE
rancher_secret_key = pvj8kTQjNgQddXxujdxeq1ht9EJfUc8G5SgdwSEY
rancher_server_ip = 18.184.148.72
rancher_server_private_dns = ip-172-31-20-237.eu-central-1.compute.internal
rancher_url = http://18.184.148.72:8080
```

CloudFormation

```
302   VirtualMachine:
303     DependsOn: EIPAssociation
304     Type: 'AWS::EC2::Instance'
305     Metadata:
306       'AWS::CloudFormation::Init':
307         configSets:
308           default: !If [HasIAMUserSSHAccess, [awslogs, ssh-access, config], [awslogs, config
309         awslogs:
310           packages:
311             yum:
312               awslogs: []
313           files:
314             '/etc/awslogs/awscli.conf':
315               content: !Sub |
316                 [default]
317                 region = ${AWS::Region}
318                 [plugins]
319                 cwlogs = cwlogs
320               mode: '000644'
321               owner: root
322               group: root
```

```
574 Outputs:
575   TemplateID:
576     Description: 'cloudonaut.io template id.'
577     Value: 'ec2/ec2-auto-recovery'
578   TemplateVersion:
579     Description: 'cloudonaut.io template version.'
580     Value: '__VERSION__'
581   StackName:
582     Description: 'Stack name.'
583     Value: !Sub '${AWS::StackName}'
584   InstanceId:
585     Description: 'The EC2 instance id.'
586     Value: !Ref VirtualMachine
587     Export:
588       Name: !Sub '${AWS::StackName}-InstanceId'
589   IPAddress:
590     Description: 'The public IP address of the EC2 instance.'
591     Value: !Ref ElasticIP
592     Export:
593       Name: !Sub '${AWS::StackName}-IPAddress'
594
```

Pulumi

```
const aws = require("@pulumi/aws");

let size = "t2.micro";
let ami  = "ami-0233214e13e500f77";

let group = new aws.ec2.SecurityGroup("webserver-secgrp", {
    ingress: [
        { protocol: "tcp", fromPort: 22, toPort: 22, cidrBlocks: ["0.0.0.0/0"] },
    ],
});

let server = new aws.ec2.Instance("webserver-www", {
    instanceType: size,
    securityGroups: [ group.name ],
    ami: ami,
});

exports.publicIp = server.publicIp;
exports.publicHostName = server.publicDns;

```

```
florian@Florians-MBP ~/g/s/g/h/pulumi-hello-world> pulumi stack output
Current stack outputs (2):
    OUTPUT                                      VALUE
    publicHostName                              ec2-54-93-33-7.eu-central-1.compute.amazonaws.com
    publicIp                                    54.93.33.7
```

```
Performing changes:
* pulumi:pulumi:Stack: (same)
    [urn=urn:pulumi:pulumi-hello-world-dev::pulumi-hello-world::pulumi:pulumi:Stack::pulumi-hello-world-pulumi-hello-world-dev]
    ~ aws:ec2/securityGroup:SecurityGroup: (update)
        [id=sg-0c0253835c1d9b904]
        [urn=urn:pulumi:pulumi-hello-world-dev::pulumi-hello-world::aws:ec2/securityGroup:SecurityGroup::webserver-secgrp]
        description        : "Managed by Pulumi"
      ~ ingress            : [
            [0]: {
                    cidrBlocks: [
                        [0]: "0.0.0.0/0"
                    ]
                    fromPort  : 22
                    protocol  : "tcp"
                    self      : false
                    toPort    : 22
                }
          + [1]: {
                  + cidrBlocks: [
                  +     [0]: "0.0.0.0/0"
                    ]
                  + fromPort  : 80
                  + protocol  : "tcp"
                  + self      : false
                  + toPort    : 80
                }
        ]
```

Resources Pulumi

- Pulumi Concept: https://bit.ly/2N232VH
- Hacker News: https://bit.ly/2O6sm9y
- Slides: https://oreil.ly/2N2IPiH
- Programming The Cloud: https://bit.ly/2wZv9KD

Appendix B

Custom CI/CD Pipeline with DroneCI

TODO