ANSIBLE

Serverless + Ansible

Ryan Scott Brown
Senior Software Engineer, Ansible Core

redhat

ANSIBLE

~~Serverless~~ Advanced Cloud + Ansible

Ryan Scott Brown
Senior Software Engineer, Ansible Core

redhat

Agenda

- Bias review
- Ansible Philosophy
- Serverless Overview
  - Why Serverless?
- Ansible for Cloud Resources
- Fake hosts, roles, and practices
- Deployments with Ansible
  - Serverless Framework & Ansible
  - All Ansible
  - CloudFormation & Ansible
- Coming Soon
  - AWS SAM
  - More modules!

Biases

- I work for Ansible
- Public cloud fan
- Primarily focused on Linux infra

Ansible Philosophy

Automation for the People

Simple Wins

- Simple, powerful automation
- Play well with others
- Work for users
- Listen to the community

Cloud for the People

- Modules to cover different providers
- Dynamic inventory for changing resources
- Smart credential handling
- Split abstraction
  - Control-plane “on the cloud”
  - Instance level “in the cloud”
- Bring tools you already use

Serverless

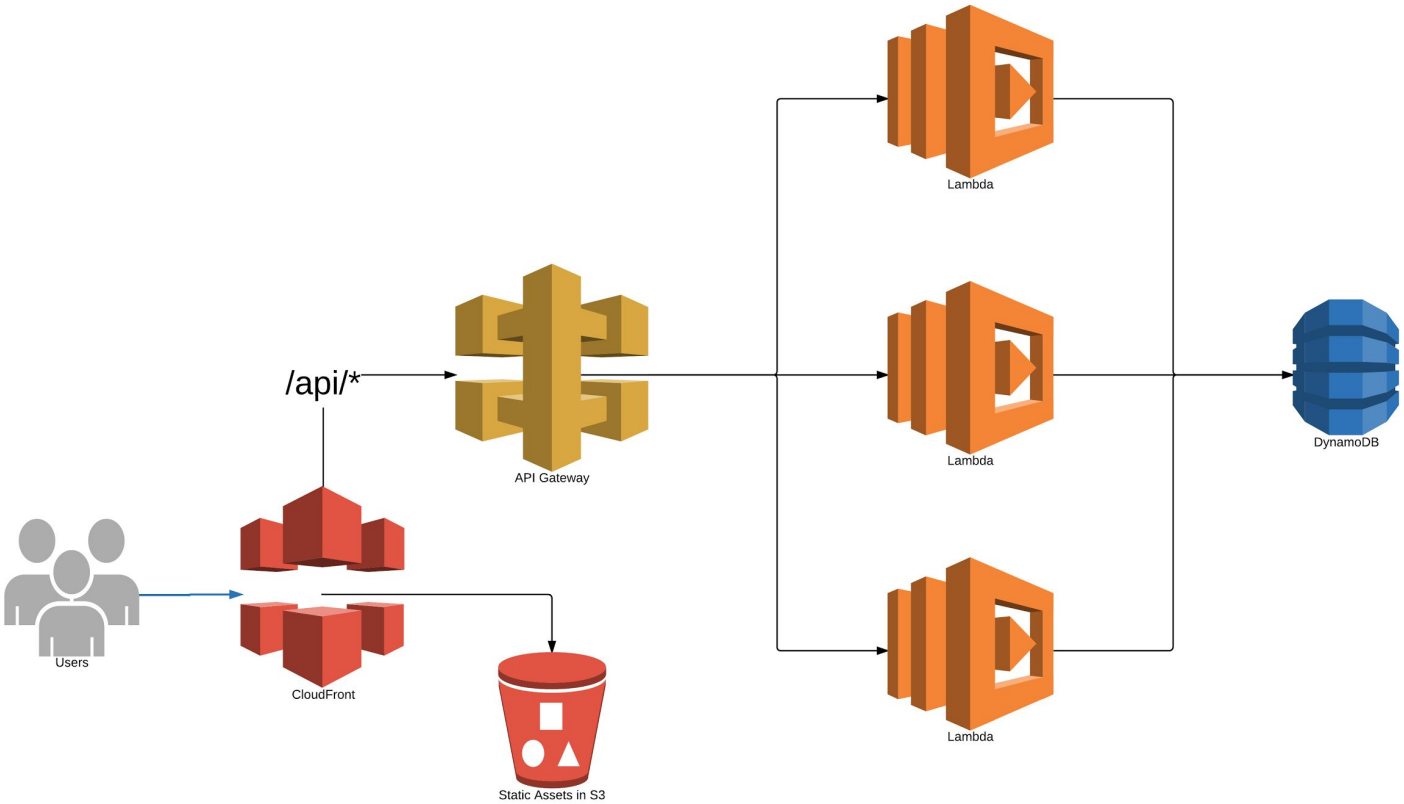
So I don’t need orchestration, right?

For Developers

- Pay per use with fine-grained metering
- Autoscaling driven by demand
- Invocations create their own new environment
- Fewer barriers to new development

For Operators

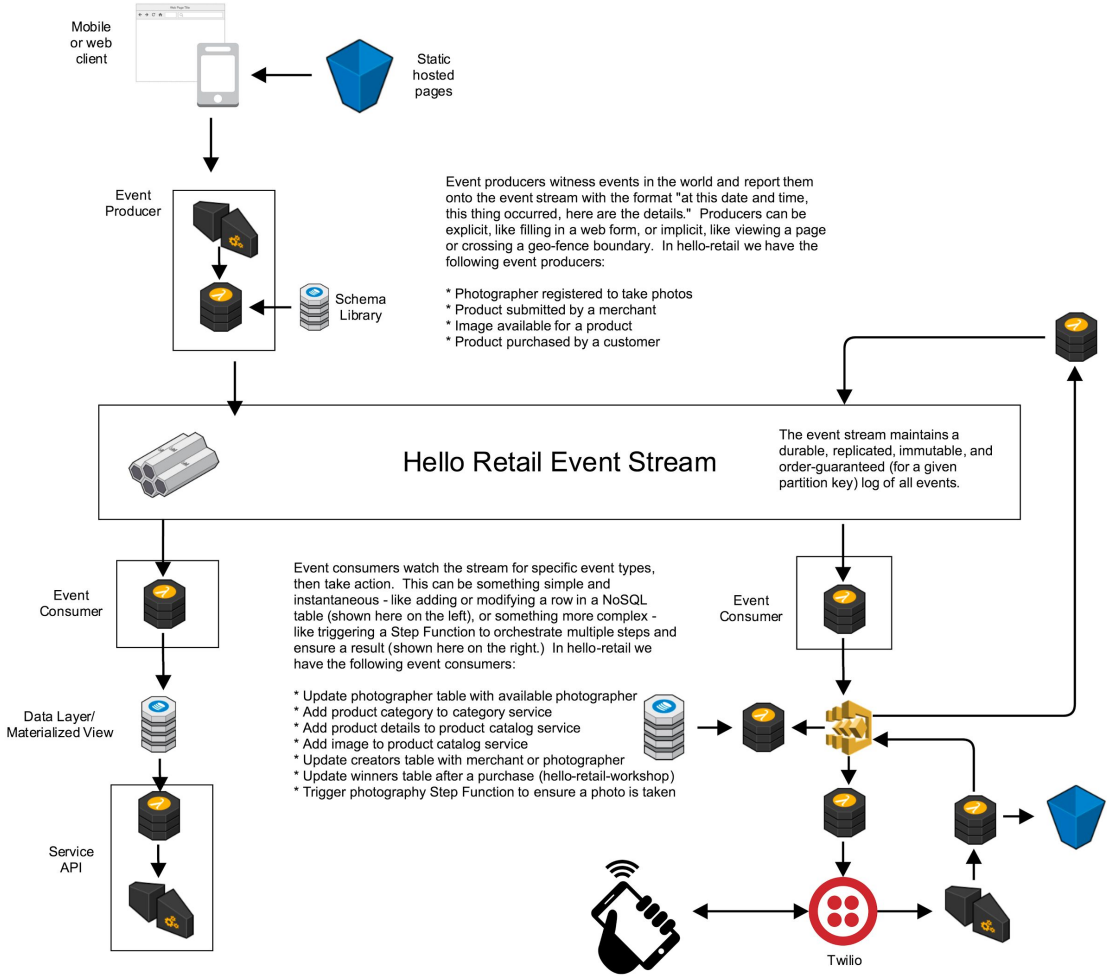
- Autoscaling driven by demand
- Everything is behind an API
- Consistent, repeatable environment
- Dev/Prod parity


Lambda
/api/*
API Gateway
Lambda
DynamoDB
Users
CloudFront
Static Assets in S3
Lambda

AWS Lambda, API Gateway, DynamoDB, and Friends*

- Narrowly focused tools connected via APIs
- Replace custom development with configuration
- Free up people to spend more time on other tasks

*And plenty of competitors and frenemies



Source: https://github.com/Nordstrom/hello-retail

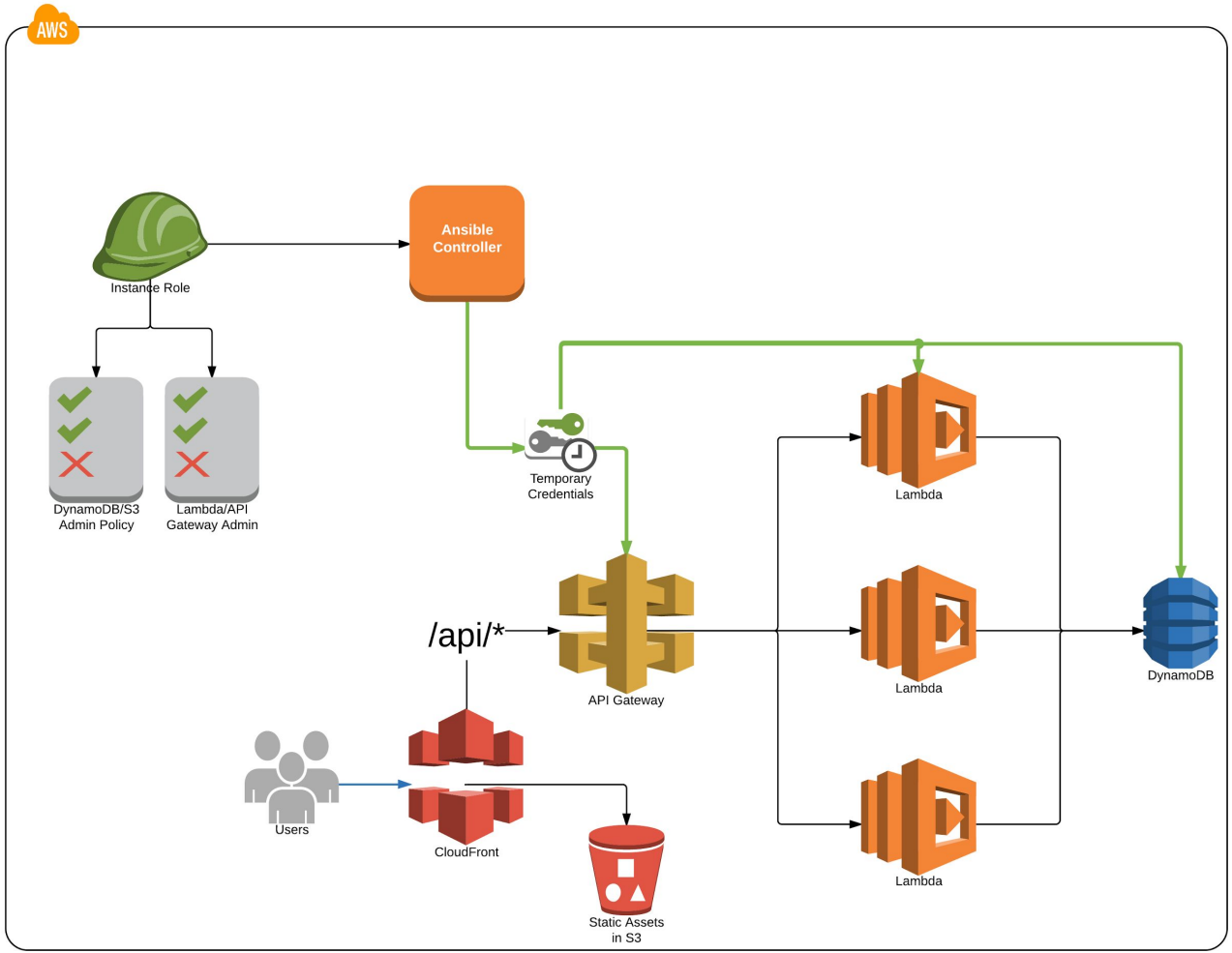
Where is Ansible?

- Ansible control can be done from anywhere with API access
  - Tower in your datacenter or EC2
  - Remote node in EC2
- Credentialed as an IAM user or instance role
- Serverless solutions trade custom components for services
  - More varied APIs
  - More configuration spread across services

AWS Lambda, API Gateway, DynamoDB, and Friends*

- Narrowly focused tools connected via **APIs**
- Replace custom development with **configuration**
- Free up people to spend more time on other tasks

*And plenty of competitors and frenemies


AWS
Instance Role
Ansible Controller
DynamoDB/S3 Admin Policy
Lambda/API Gateway Admin
Temporary Credentials
Lambda
Lambda
Lambda
DynamoDB
/api/*
API Gateway
Users
CloudFront
Static Assets in S3

Ansible Support

- aws_api_gateway
- data_pipeline
- execute_lambda
- lambda
- lambda_policy
- lambda_event
- cloudwatchevent_rule
- dynamodb_table
- dynamodb_ttl
- elasticache
- azure_rm_functionapp

But the Inventory!

Hostless?

Virtual Hosts for Service Settings

- Supports group+host variables
- Groups are reusable for host orchestration
- Minimizes vars in the global namespace

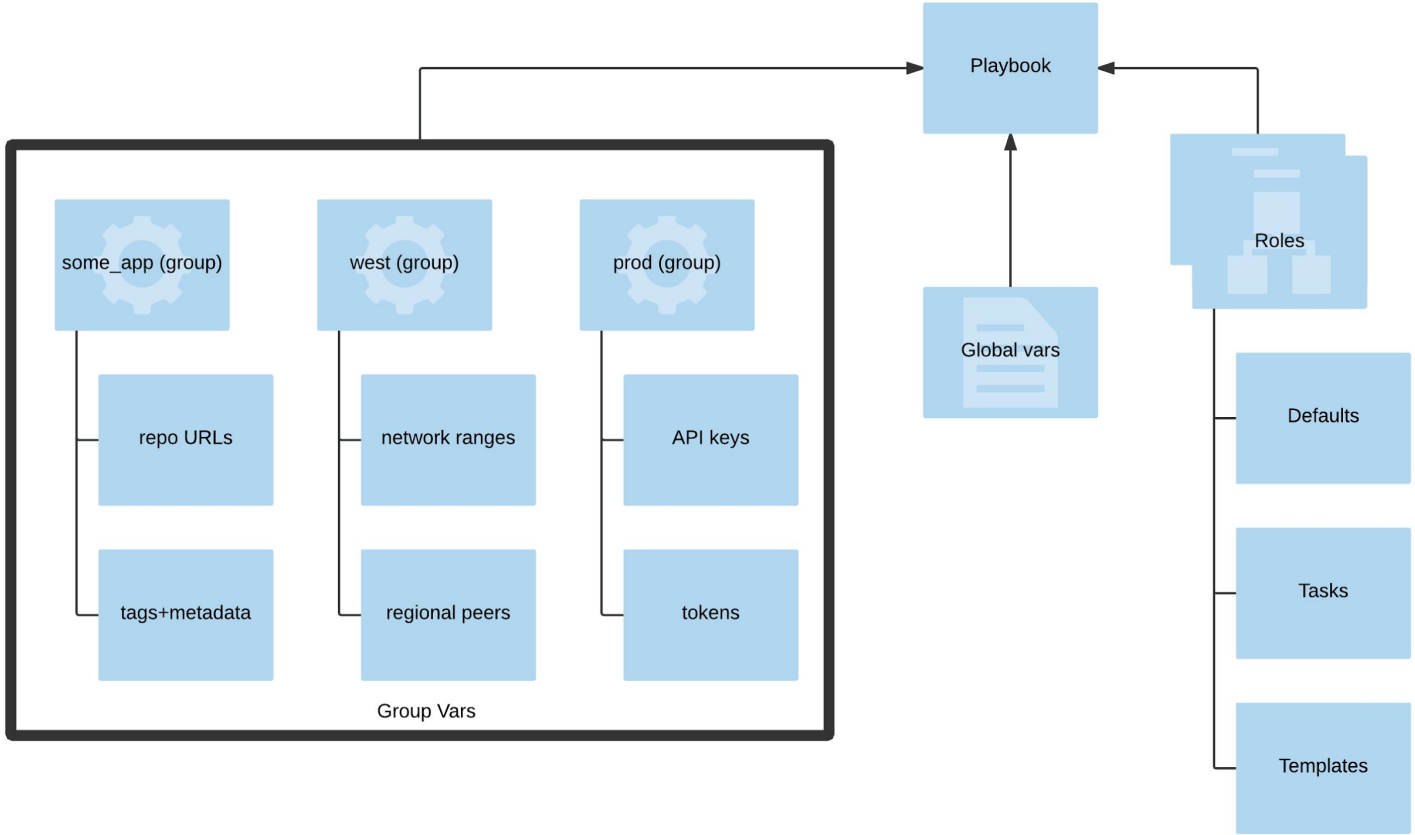
Managing Configuration (Inventory)


```
sls_east_prod ansible_connection=local ansible_host=127.0.0.1
sls_west_prod ansible_connection=local ansible_host=127.0.0.1
sls_west_dev ansible_connection=local ansible_host=127.0.0.1
sls_east_dev ansible_connection=local ansible_host=127.0.0.1

[prod]
sls_east_prod
sls_west_prod

[east]
sls_east_prod
sls_east_dev
```

Managing Configuration (Variables)


```
region: us-east-1
azs:
- us-east-1a
- us-east-1b
- us-east-1c
- us-east-1d
- us-east-1e
peers:
  db: e1-replica-sql-{{ stage }}.{{ tld }}
tld: east.internal.myco.com.
```

Managing Configuration (Variables)



Roles for Service Settings

- Use role defaults and override as needed
- Sharable between teams for common tooling
  - CloudFormation custom resources
  - Evented security/auditing functions
  - Lambda extensions to AutoScaling or ECS
- Role namespace keeps serverless-specific vars off instances

Implementation Patterns

Pattern 1

Ansible and Serverless Framework

https://serverless.com/framework

Ansible & Serverless

Best of Both Worlds

- Toolkit for (cross-provider) serverless apps
- Unified CLI and config
- Community focused exclusively on serverless tools

Ansible & Serverless

Why both?

- Helps to have devs with Serverless Framework experience
- Can pass control for shared resources up to Ansible

Cross-Cloud Support

- AWS Lambda
- Azure Functions
- Google Cloud Functions
- OpenWhisk
- Kubeless
- SpotInst

Ansible & Serverless


```
- serverless_deploy:
    stage: prod
    functions:
    - createUser
    - deleteUser
    region: us-east-1
```

Migration Tips

- Attention to detail - who owns resources needed by both?
- Make sure data is close enough to new & old systems
- Remember Ansible can populate Serverless variable files

Exporting Variables

```
- name: Save out variables to YAML for sls framework
  copy:
    content: “{{ sls_vars | to_nice_yaml }}”
    dest: “{{ project_dir }}/vars/ansible-generated.yml”
```

Exported Serverless Vars

```
lambda_iam_arn: arn:aws:iam:.......:role/MyExecRole
shared_kinesis: some-event-stream
```

Ansible & Serverless


```
custom:
  stage: ${opt:stage, self:provider.stage, self:custom.private.stage}
  private: ${file(../secrets.yml)}
  external: ${file(vars/ansible-generated.yml)}

provider:
  name: aws
  runtime: python3.6
  environment:
    STAGE: ${self:custom.stage}

functions:
  eventWriter:
    role: ${self:custom.external.lambda_iam_arn}
    handler: eventWriter.ingest
    environment:
      STREAM_NAME:
        Fn::ImportValue: ${self:custom.exportStreamName}
    events:
      - http:
          path: event-writer
          method: post
```

Pattern 2

Ansible All-In

Ansible Only

Fastest Start

- Ansible skills 100% transferrable
- Add to existing playbooks
- Great for mixed workloads

Build a Function

```
- lambda:
    name: sendReportMail
    zip_file: "{{ deployment_package }}"
    runtime: python2.7
    timeout: 20
    handler: handler.handler
    memory_size: 1024
    role: "{{ iam_exec_role }}"
  register: lambda
```

Add HTTPS Endpoint (New in 2.4)

```
- aws_api_gateway:
    api_file: my-swagger-def.yml
    stage: dev
```

Ansible Only


```
swagger: "2.0"
basePath: "/something"
paths:
  /:
    get:
      consumes:
      - "application/json"
      produces:
      - "application/json"
      responses:
        200:
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
```

Cron Scheduling

```
- cloudwatchevent_rule:
    description: "Send out incident summary daily"
    name: incident_summary_schedule
    schedule_expression: 'rate(1 day)'
    targets:
      - id: SendSummary
        arn: "{{ lambda.configuration.function_arn }}"
        input: '{}’
```

Pattern 3

Ansible + CloudFormation

Self-Contained Apps

- If you have CloudFormation skills already
- AWS publishes reference architectures
- Check mode now uses full ChangeSets
- Serverless Application Model?

Sample Play

```
- cloudformation:
    stack_name: “{{ env }}-sls-app”
    state: present
    template: sls_app.yml
    template_parameters:
      Environment: “{{ env }}”
      …
- cloudformation_facts:
    stack_name: “{{ env }}-sls-app”
```

Ansible & CloudFormation


```
- debug:
    var: cloudformation[“{{ env }}-sls-app”].outputs
```

```
[{
    "description": "Current function version",
    "output_key": "FuncQualifiedArn",
    "output_value": "arn:aws:lambda:...:function:MyFunc:3"
}]
```

Coming Soon

Ansible 2.5 and beyond

AWS Serverless Application Model

- Extensions for CloudFormation server-side transforms
- Rolls packaging/uploading step into stack actions
- Creates a shorthand language for Lambda and API Gateway

Soon - AWS SAM


```
Transform: AWS::Serverless-2016-10-31
Resources:
  CreateThumbnail:
    Type: AWS::Serverless::Function
    Properties:
      Handler: handler
      Runtime: runtime
      Timeout: 60
      Policies: AWSLambdaExecute
      Events:
        CreateThumbnailEvent:
          Type: S3
          Properties:
            Bucket: !Ref SrcBucket
            Events: s3:ObjectCreated:*
```

Sample Play

```
- cloudformation:
    stack_name: sam-test-app
    state: present
    project_path: “{{ somewhere }}”
    template: sam_template.yml
- cloudformation_facts:
    stack_name: sam-test-app
```

Thank You!

Questions*?

Tweet @ryan_sb or find me at the social tonight

*Maybe right now, time permitting

ANSIBLE

Serverless