Automation with Ansible and EC2

Michael Menzel
Vineet Hingorani
September 02, 2015

Agenda

Recap: Cloud Computing & Compute Clouds

- Definition, Characteristics, Delivery Models, Core Concepts

Infrastructure Automation with OSCM

- Concept of OSCM, Opscode Chef, Saltstack Salt

Ansible

- Explain the Ansible Approach, Share our Experiences

Examples with Ansible

- Start Example App on multiple Instances with Load Balancer, Start MetricStorageService and use Auto-Scaling

Outlook

- Ansible Tower, Discussion on how to leverage

Recap: Cloud Computing

Cloud Computing Definition

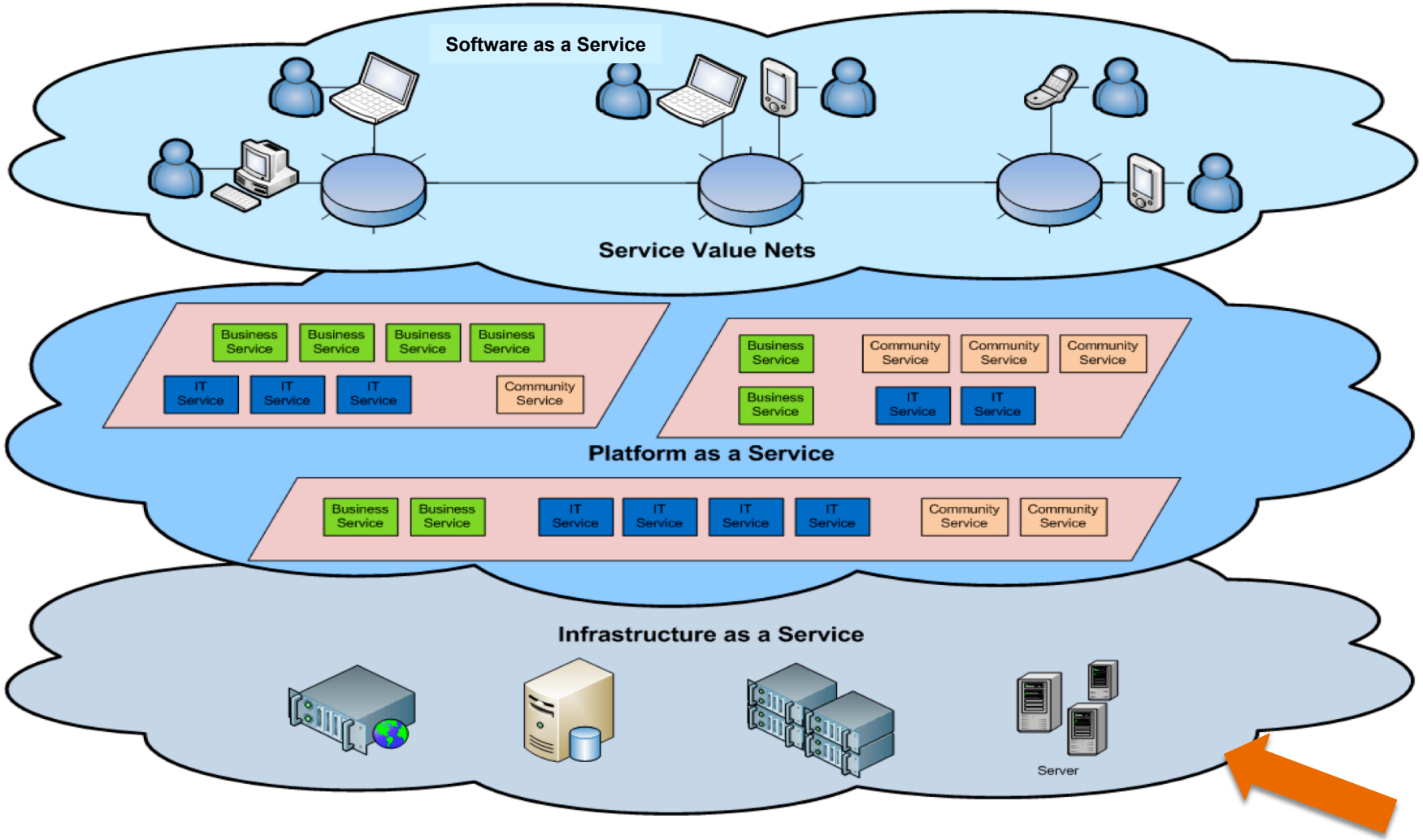
Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released ***with minimal management effort or service provider interaction.***

Source: National Institute for Standards & Technology (NIST)
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Cloud Computing Characteristics

Relevant to this talk

Cloud Delivery Models



SaaS
Web Applications like
Google Apps, Salesforce etc.

PaaS
Development or execution environments like
Google App Engine, Windows Azure

IaaS
Offerings like Google Storage, Amazon Web Services etc.

Focus of this talk

Amazon Web Services

Amazon Web Services

Compute
- **EC2** — Virtual Servers in the Cloud
- **EC2 Container Service** — Run and Manage Docker Containers
- **Elastic Beanstalk** — Run and Manage Web Apps
- **Lambda** — Run Code in Response to Events

Storage & Content Delivery
- **S3** — Scalable Storage in the Cloud
- **CloudFront** — Global Content Delivery Network
- **Elastic File System** PREVIEW — Fully Managed File System for EC2
- **Glacier** — Archive Storage in the Cloud
- **Storage Gateway** — Integrates On-Premises IT Environments with Cloud Storage

Database
- **RDS** — MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
- **DynamoDB** — Predictable and Scalable NoSQL Data Store
- **ElastiCache** — In-Memory Cache
- **Redshift** — Managed Petabyte-Scale Data Warehouse Service

Networking
- **VPC** — Isolated Cloud Resources
- **Direct Connect** — Dedicated Network Connection to AWS
- **Route 53** — Scalable DNS and Domain Name Registration

Developer Tools
- **CodeCommit** — Store Code in Private Git Repositories
- **CodeDeploy** — Automate Code Deployments
- **CodePipeline** — Release Software using Continuous Delivery

Management Tools
- **CloudWatch** — Monitor Resources and Applications
- **CloudFormation** — Create and Manage Resources with Templates
- **CloudTrail** — Track User Activity and API Usage
- **Config** — Track Resource Inventory and Changes
- **OpsWorks** — Automate Operations with Chef
- **Service Catalog** — Create and Use Standardized Products

Security & Identity
- **Identity & Access Management** — Manage User Access and Encryption Keys
- **Directory Service** — Host and Manage Active Directory
- **Trusted Advisor** — Optimize Performance and Security

Analytics
- **EMR** — Managed Hadoop Framework
- **Data Pipeline** — Orchestration for Data-Driven Workflows
- **Kinesis** — Real-time Processing of Streaming Big Data
- **Machine Learning** — Build Smart Applications Quickly and Easily

Mobile Services
- **Cognito** — User Identity and App Data Synchronization
- **Device Farm** — Test Android, Fire OS, and iOS apps on real devices in the Cloud
- **Mobile Analytics** — Collect, View and Export App Analytics
- **SNS** — Push Notification Service

Application Services
- **API Gateway** — Build, Deploy and Manage APIs
- **AppStream** — Low Latency Application Streaming
- **CloudSearch** — Managed Search Service
- **Elastic Transcoder** — Easy-to-use Scalable Media Transcoding
- **SES** — Email Sending Service
- **SQS** — Message Queue Service
- **SWF** — Workflow Service for Coordinating Application Components

Enterprise Applications
- **WorkSpaces** — Desktops in the Cloud
- **WorkDocs** — Secure Enterprise Storage and Sharing Service
- **WorkMail** PREVIEW — Secure Email and Calendaring Service

EC2 is the elastic compute cloud of AWS offering “computers” as a service.

EC2 Core Concepts

Amazon Machine Image (AMI)

An encrypted file stored in Amazon storage, containing all the information necessary to boot instances of a customer's software

- An AMI is like a bootable root disk, which can be pre-defined or user-built.
  - Public AMIs: Pre-configured, template AMIs. Eg. Ubuntu 14.04, Red Hat Enterprise 7
  - Private AMIs: User-built AMI containing private applications, libraries, data and associated configuration settings. Eg. Ubuntu 14.04 image with HANA as a software.
- There is a community of AMI creators, e.g. bitnami.com, thecloudmarket.com, aws.amazon.com/marketplace

Instance

The running system based on an AMI

- All instances based on the same AMI begin executing identically. An instance can be launched in a few minutes.
- Any information on them is lost when the instances are terminated or if they fail.
- Data can be made persistent by use of Amazon storage services.

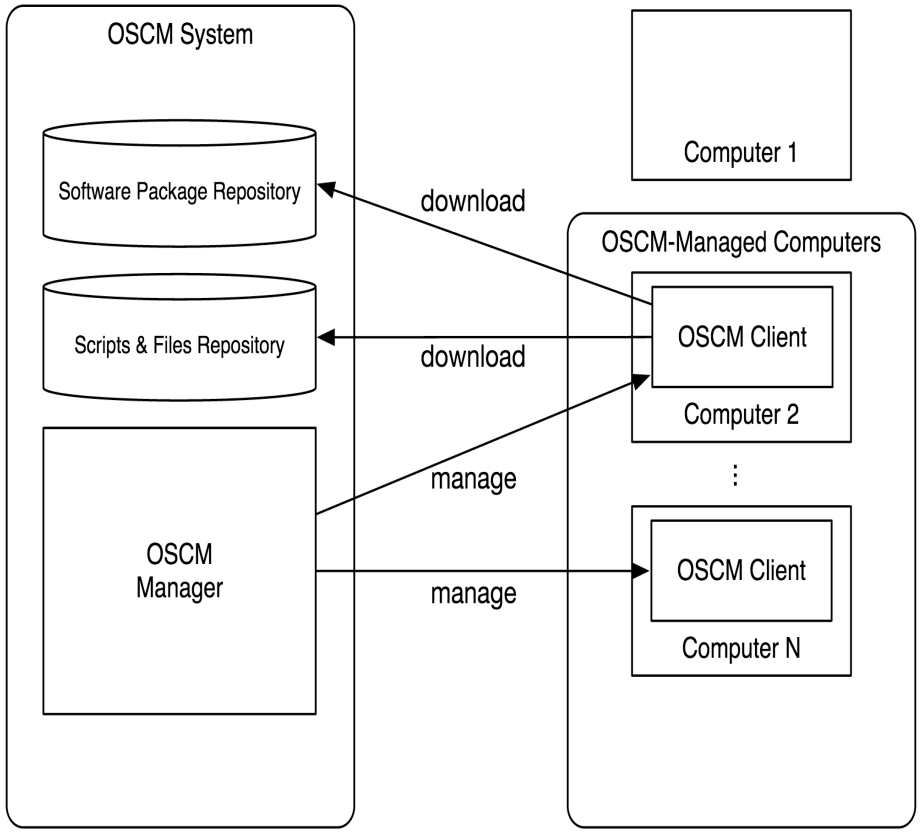
Infrastructure Automation

Infrastructure Automation Approaches

Scripting Environments

Applying configuration to nodes (single or hundreds)

- Includes automating everything
  - Installing OS
  - Configuring servers on instances
  - Configuring how different softwares communicate with each other
- Setting up and replicating environments in less error-prone manner
  - Automated testing
  - Documentation
  - Versioning of infrastructure setup



OpsCode Chef

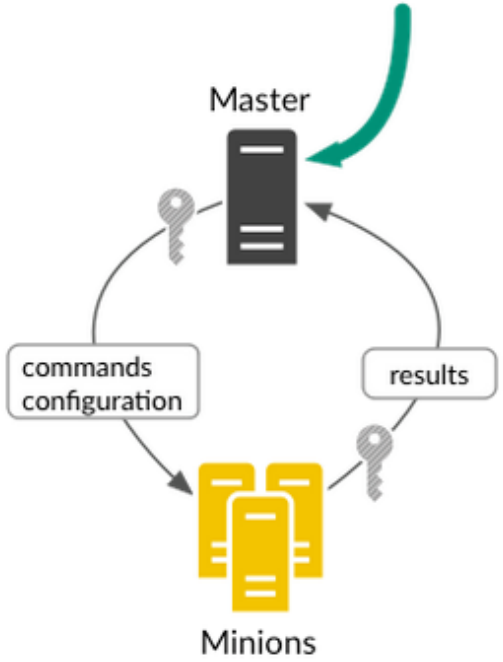
Configuration Management Tool

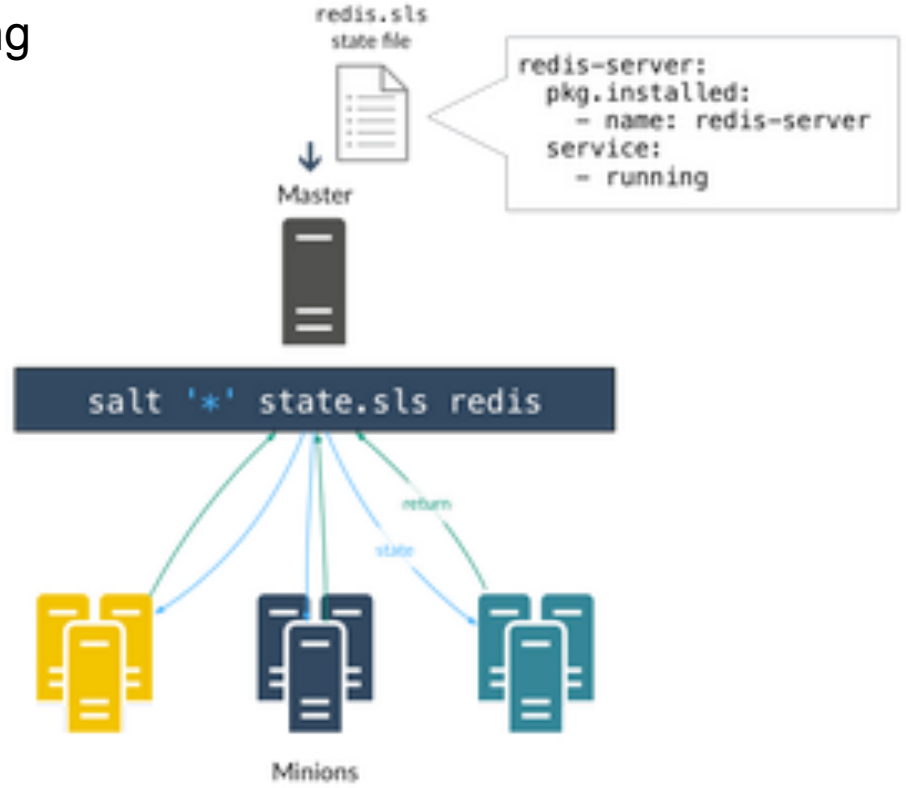
- Uses Ruby based Domain Specific Language for writing recipes
- Can integrate with cloud-based platform but not originally built for that
- The API is written both in Ruby and Erlang
- Supports multiple platforms for clients Solaris, FreeBSD, OS X, Ubuntu, Debian, Fedora, Microsoft Windows
- Server is supported on Oracle Linux/CentOS and Ubuntu

SaltStack Salt

Another Management Software for CloudOps, ITOps, DevOps

- Written in Python
- Configuration files written in YAML or PyDSL
- Master-Client architecture and needs a Master always running





Ansible

What is Ansible?

YAML based language

- Descriptive language for automation of infrastructure setups

```
YAML: true
Fruits:
- name: Apple
  price: .50
- name: Banana
  price: .80
```

Concepts

- Owns concepts to structure and reuse definition files:
  - Variables (vars)
  - Tasks
  - Profiles
  - Handlers
  - Templates (jinja2)
- Provides reusable modules to run tasks
- Can manage known hosts and cloud services

Basics

Playbooks

- Ansible's configuration, deployment and orchestration language.
- The design plan. The set of steps. The work process.

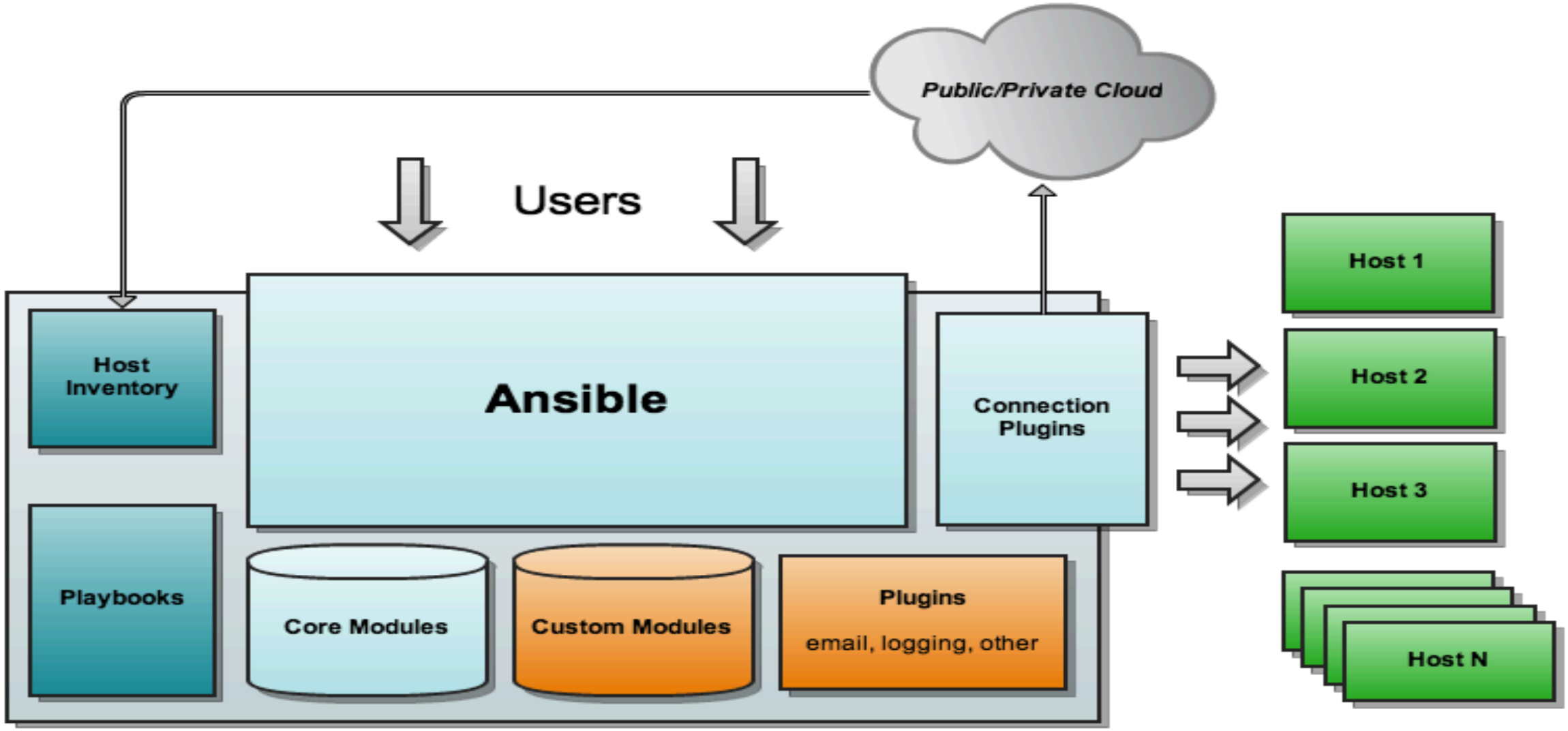
Modules

- The tasks that are actually executed in the playbooks.
- They do the actual work.
  - Core Modules
    - Maintained by the Ansible team shipped with Ansible itself.
  - Extra Modules
    - Maintained by the community and may not ship with Ansible in future.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Module yum

Ansible Setup

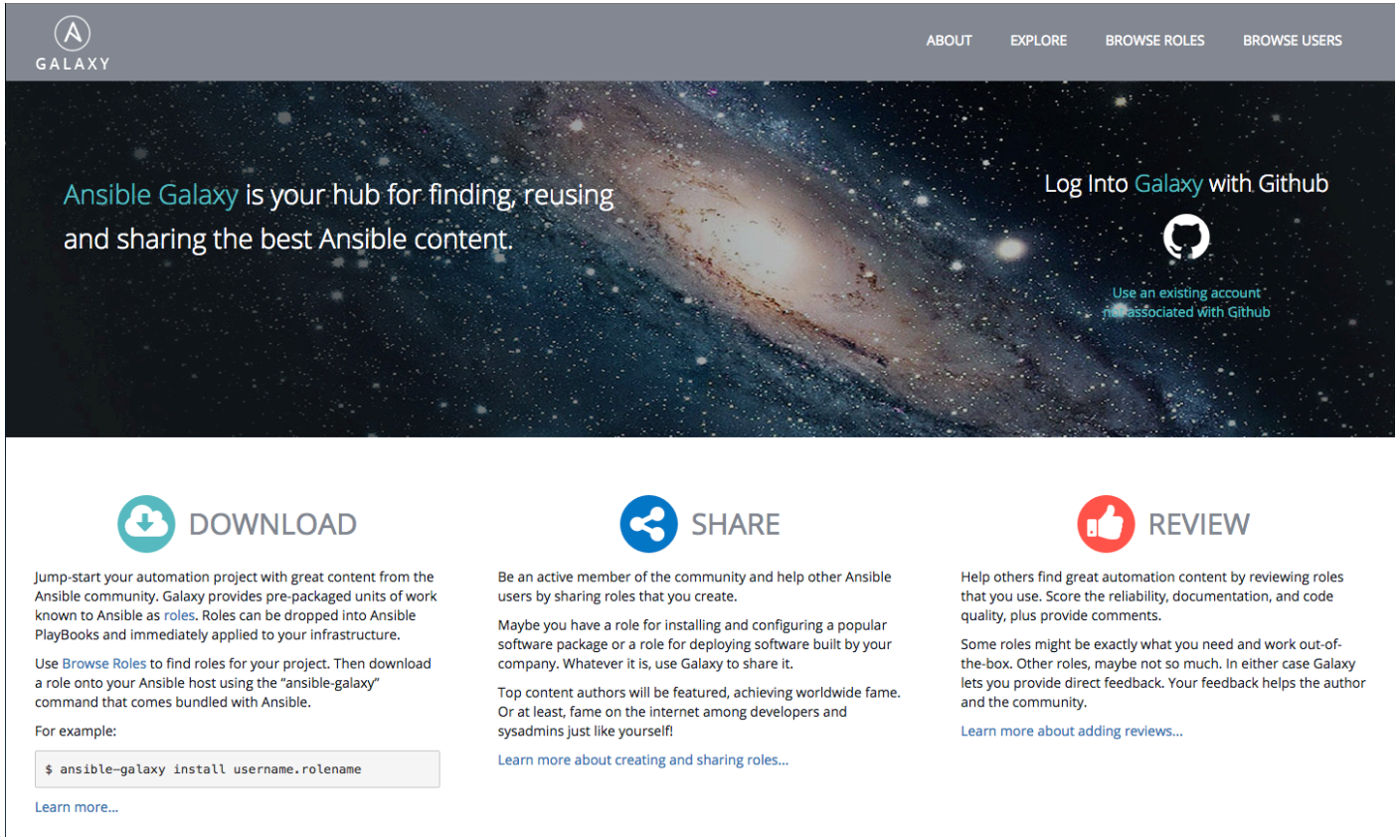


Ansible Galaxy

Ansible modules repo maintained by a big community

Many custom modules and code snippets (roles) are shared by ansible users

Also integrations with clouds, vagrant, etc.

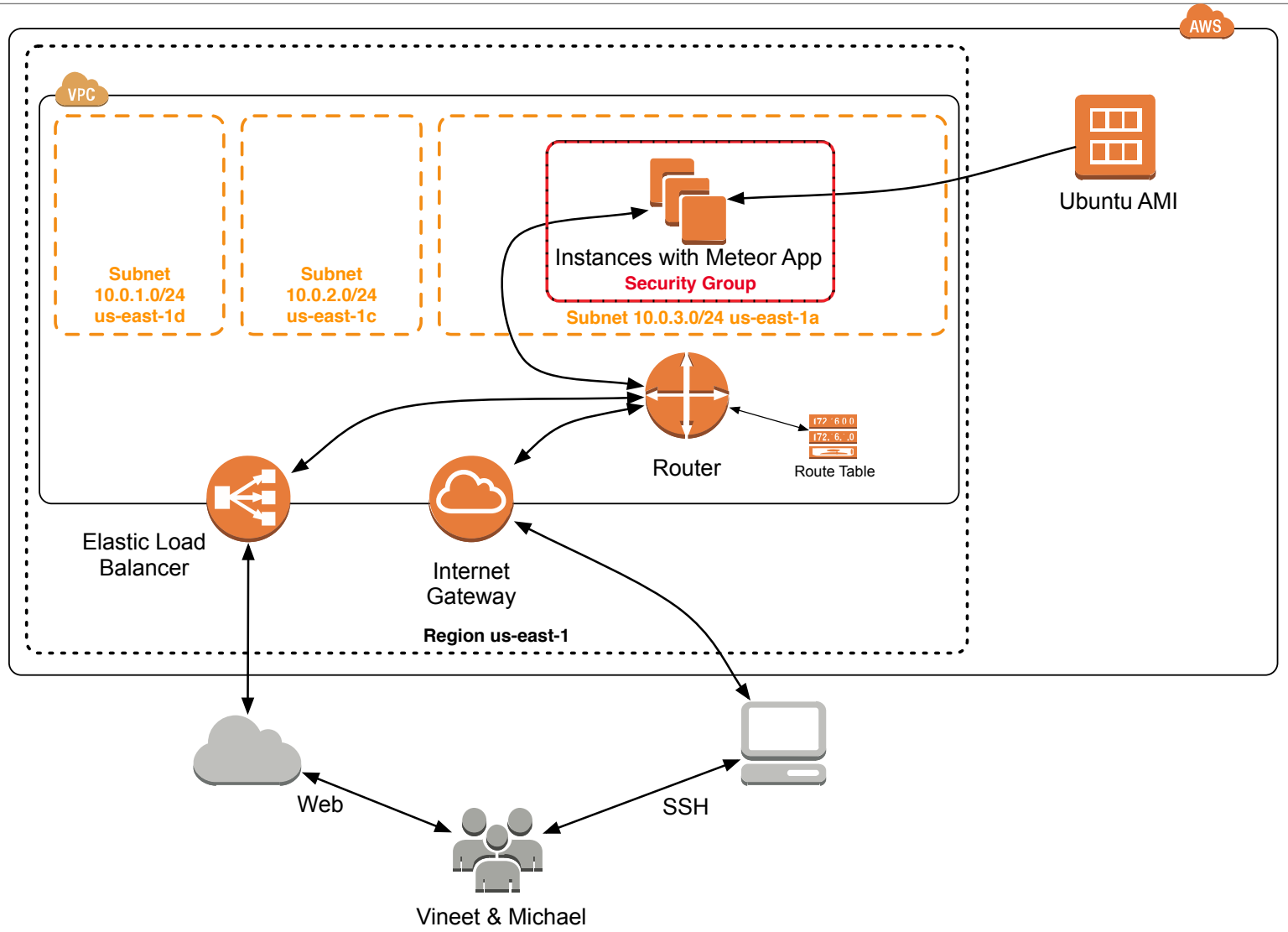


With Ansible it’s like…

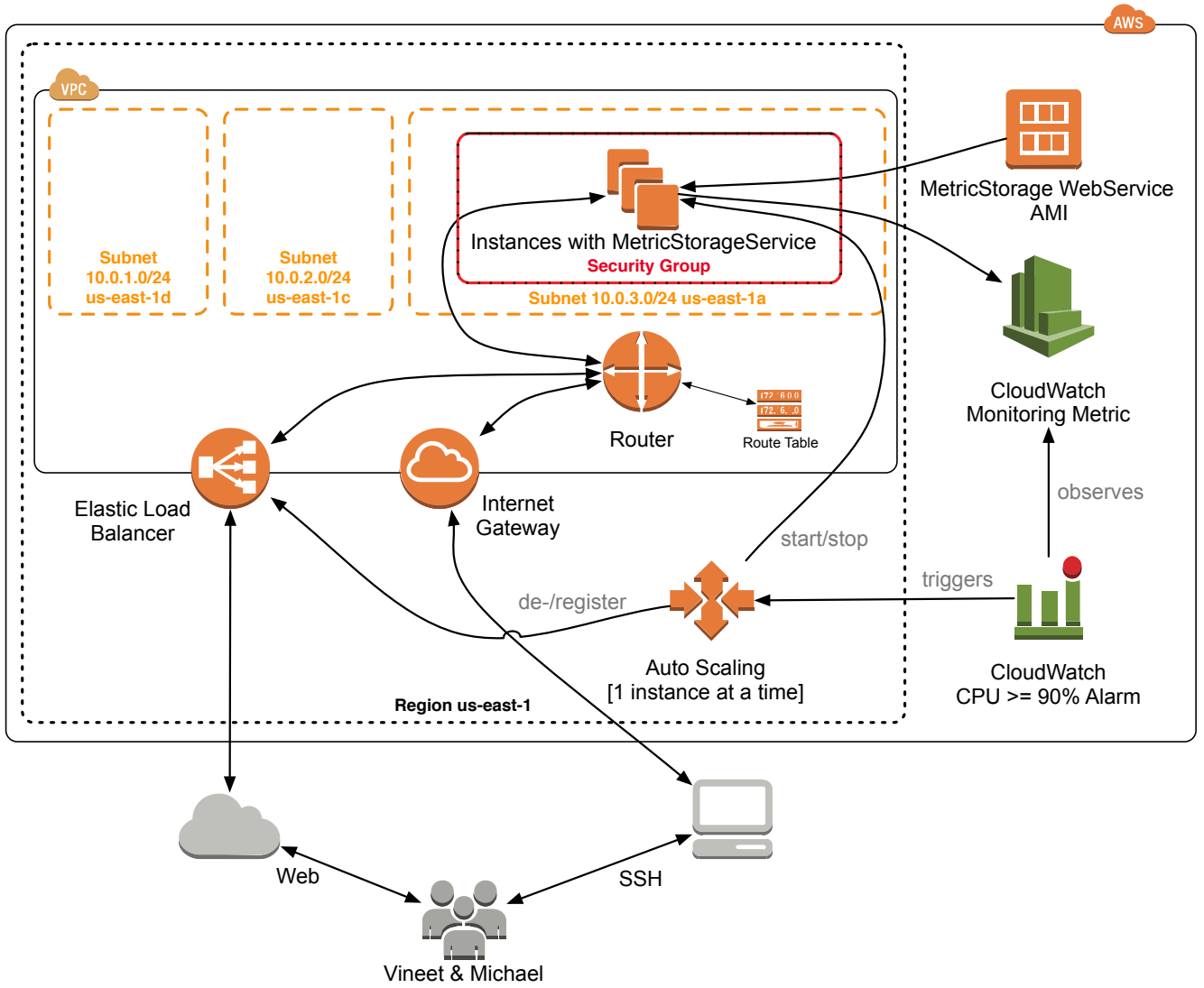
- **Defining and documenting** your infrastructure
- Thinking about a **final state** (idempotence)
- **Modules pave the way** to simple setups. Guides are available & reuse is king!
- **Iteration works** in descriptive files, too!
- No more (or less) scripting, but **still thinking**!
- Support for **continuous delivery and rolling upgrades**
- **Problems with Python or SSH** may occur (but could be with any technology)

Examples with Ansible

Ansible Example 1: Target Architecture

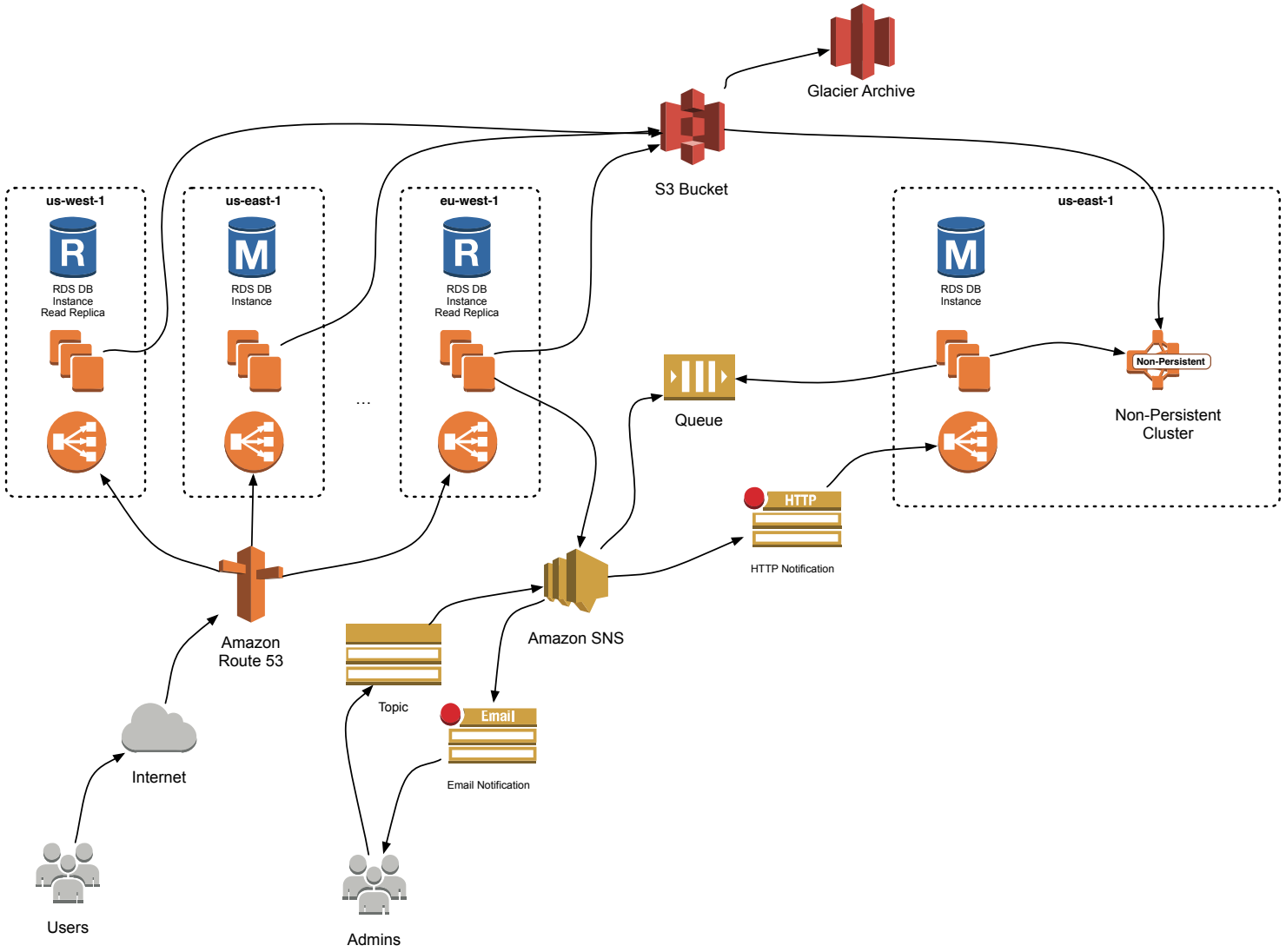


Ansible Example 2: Target Architecture



Outlook

Imagine more complex architectures



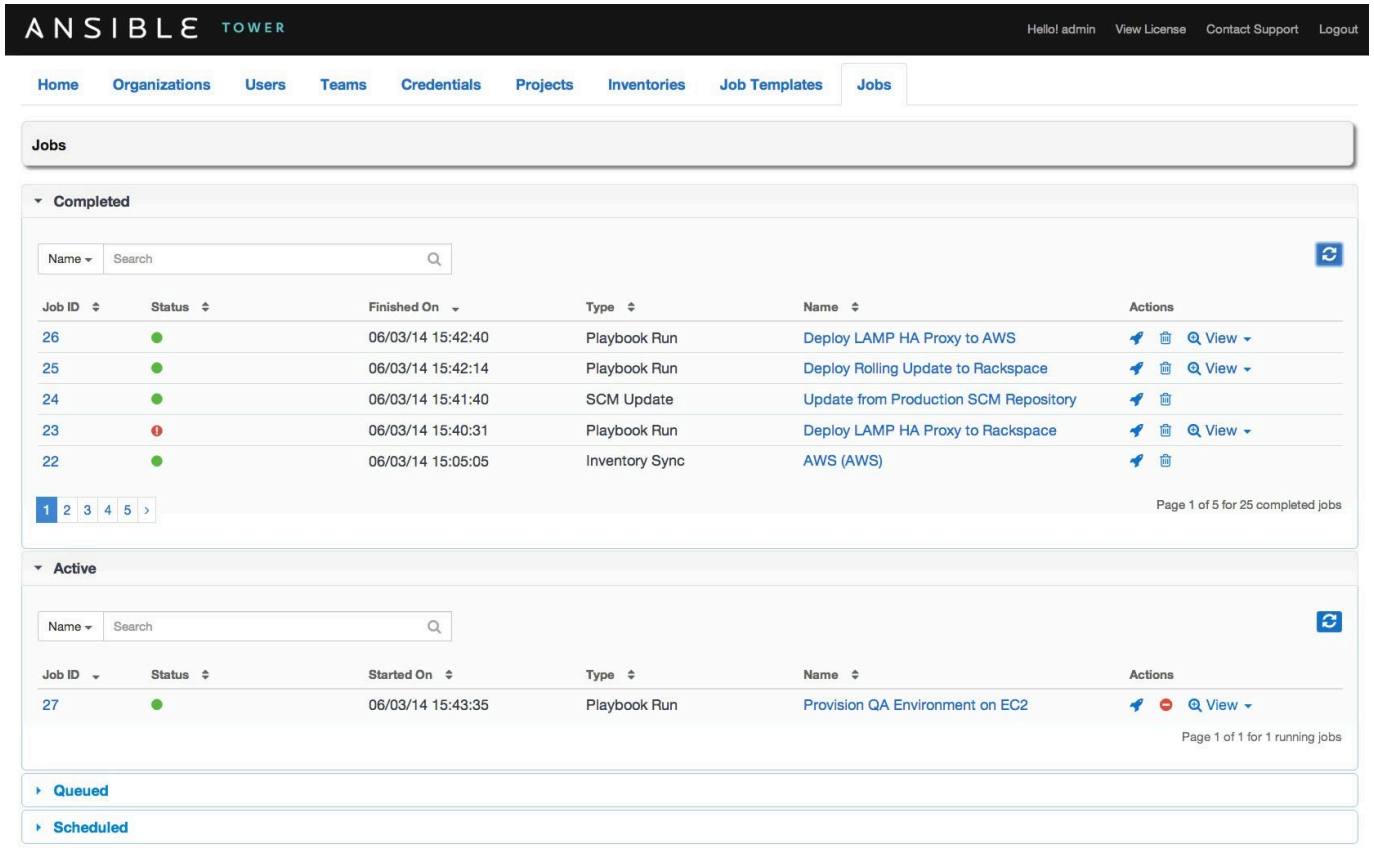
Ansible Tower

Provides a comprehensive management GUI

Costs scale with number of managed machines (good & bad)

Adds features such as:

- Scheduling of ansible tasks
- Direct machine access
- Logging of user actions



Questions?

How can we use this technology?

Thank you

Contact information:

Michael Menzel
Vineet Hingorani
C3.05