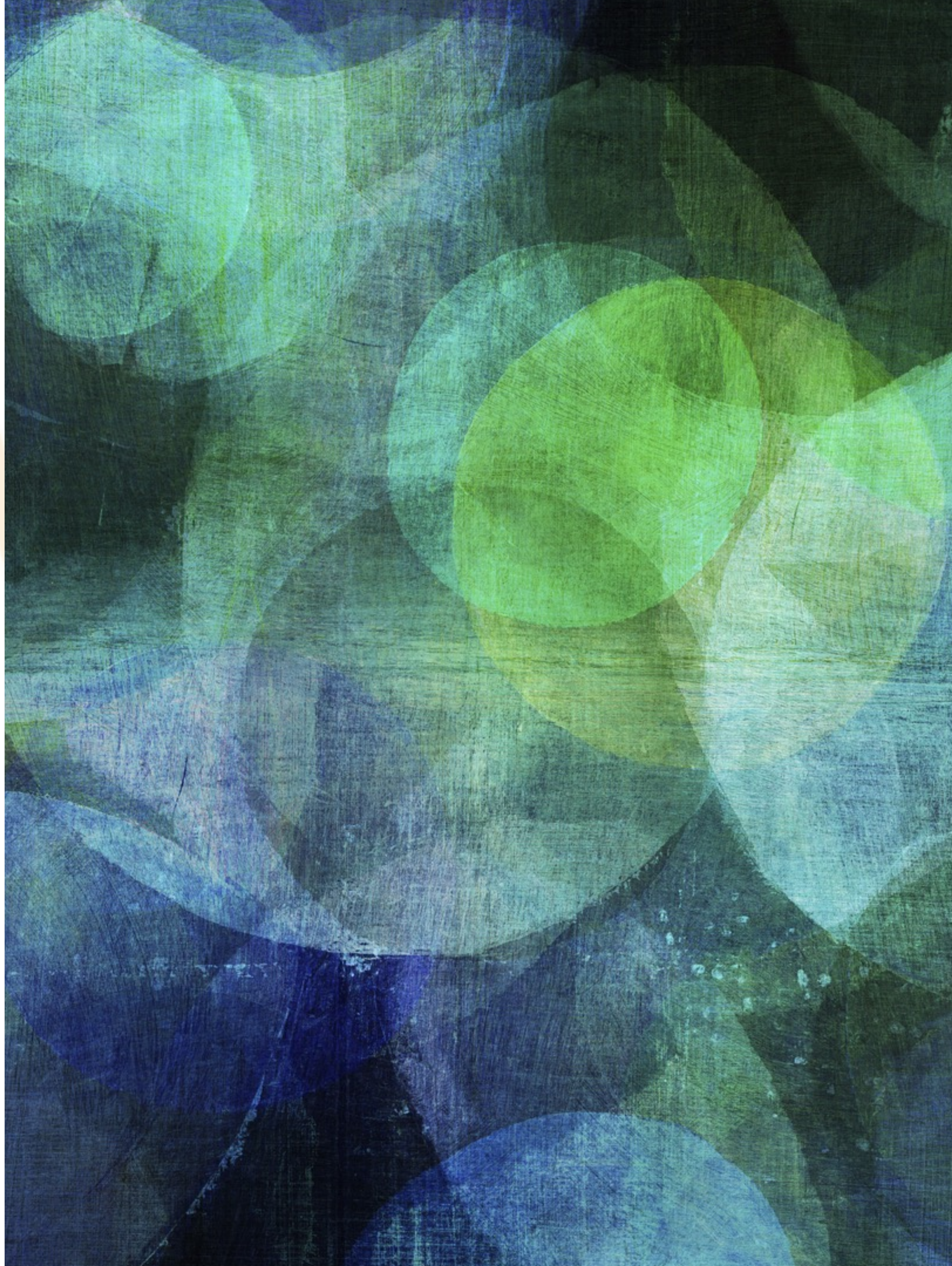
AWS DATA PROCESSING INFRASTRUCTURE 2A

Nan Dun
nan.dun@acm.org



COPYRIGHT POLICY 版权声明

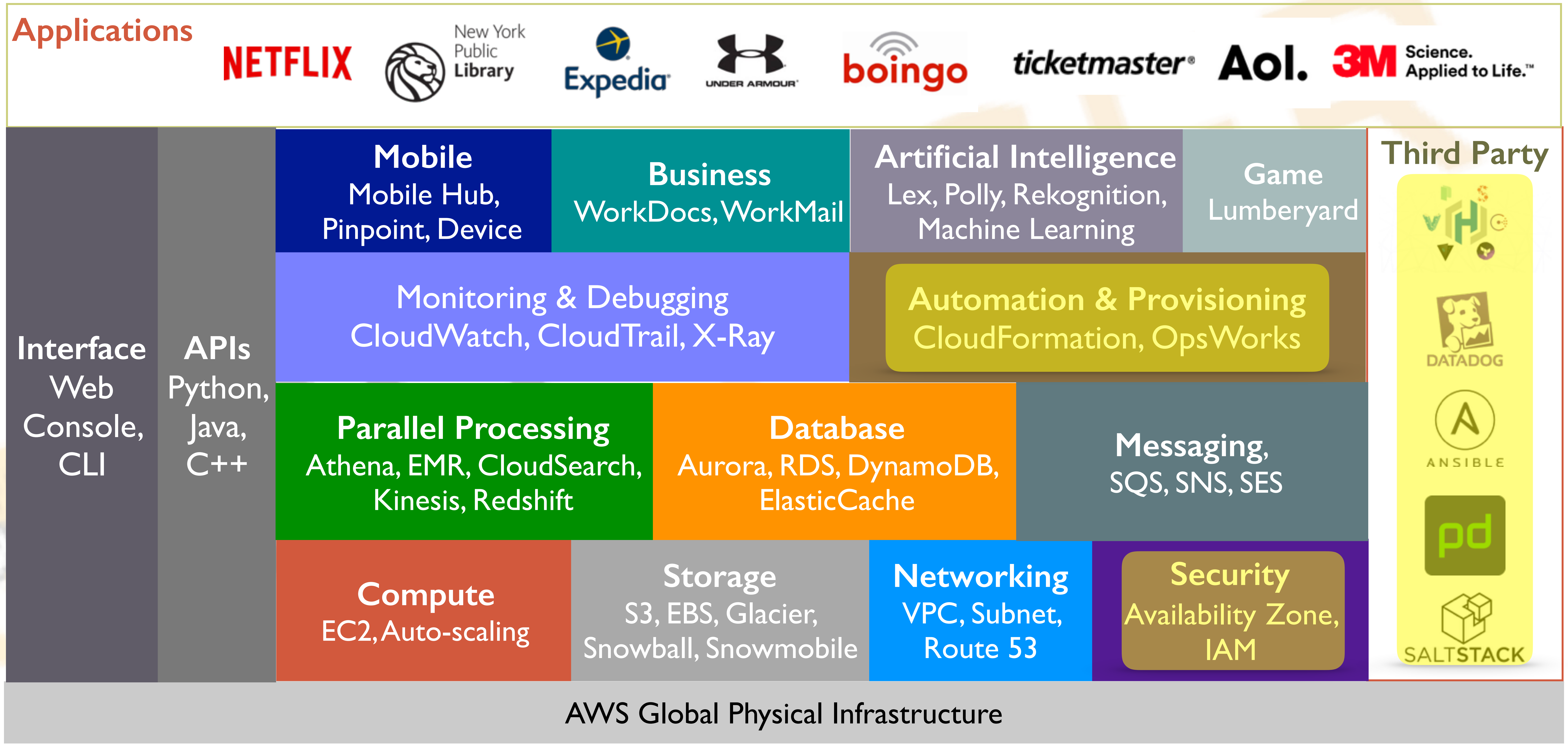
DISCLAIMER

I. "All data, information, and opinions expressed in this presentation is for informational purposes only.   I do not guarantee the accuracy or reliability of the information provided herein. This is a personal presentation. The opinions expressed here represent my own and not those of my employer."

II. "The copyright of photos, icons, charts, trademarks presented here belong to their authors."

III. "I could be wrong."

TODAY'S TOPIC



OUTLINE

- Identity and Access Management (IAM)
  - Policy, policy, policy
- Auto Provisioning
  - Packer: Build AMI
  - CloudFormation/Terraform: Infrastructure
  - Ansible: Configuration

IAM

IDENTITY AND ACCESS MANAGEMENT

- Different users shared in one single account
- Cross account access
- Fine-grained access control over all resources
- Programmable control
- Debug and find who violated the policies

AN EXAMPLE OF S3 BUCKET SHARING

- Allow all of you to
  - List the bucket
  - Get objects
  - Put objects
  - Delete objects
- Do not allow
  - Delete bucket
  - Change bucket policy

POLICY

- JSON
  - A series of statements
- Statement
  - Principal (Who)
  - Action (What)
  - Resource (Which)
  - Condition (When)

```
{
  “Statement”: [{
    “Effect”: “effect”
    “Principal”: “principal”,
    “Action”: “action”,
    “Resource”: “arn”,
    “Condition”: {
       “condition”: {
          “key”: “value”
       }}
  ]
}
```

PRINCIPALS

- Who that is allowed or denied
- Identified by ARN (Amazon Resource Name)

```
“Principal”: “AWS”: “*.*”

“Principal”: {“AWS”: “arn:aws:iam::012345678901:root”}
“Principal”: {“AWS”: “012345678901”}

“Principal”: {“AWS”: “arn:aws:iam::012345678901:user/username”}

“Principal”: {“AWS”: “arn:aws:iam::012345678901:user/username”}
“Principal”: {“AWS”: “arn:aws:iam::012345678901:role/rolename”}

“Principal”: {“Service”: “ec2.amazonaws.com”}

“Principal”: {“Federated”: “facebook.com”}
“Principal”: {“Federated”: “google.com”}
```

ACTIONS

- What that is allowed or denied

```
“Action”: “ec2:StartInstances”
“Action”: “ec2:StopInstances”

“Action”: “iam:ChangePassword”

“Action”: “s3:DeleteBucket”
“Action”: [ “s3:GetObject”, “s3:PutObject” ]

“Action”: “s3:*Bucket*”
```

Be careful to do this!

NOTACTION

- Exclude some actions from a long list != Not Allowed

```
{
  “Statement”: [{
    “Effect”: “Allow”,
    “NotAction”: “s3:*”,
    “Resource”: “*"
  }]
}
```

```
{
  “Statement”: [{
    “Effect”: “Allow”,
    “Action”: “*”,
    “Resource”: “*"
  },
  {
    “Effect”: “Deny”,
    “Action”: “s3:*”,
    “Resource”: “*"
  }]
}
```

RESOURCES

- Which resources that is applied

```
“Resources”: “arn:aws:s3::aws-nyc-taxi-data”

“Resources”: “arn:aws:ec2:us-west-2:1234567890:instance/*”

“Resources”: [
  “arn:aws:s3:::aws-nyc-taxi-data”,
  “arn:aws:s3:::aws-nyc-taxi-data/*”
]
```

CONDITIONS

- When the policy is true

```
        “Condition”: {
          “DateGreaterThan”: {“awsCurrentTime”: “2016-01-01”},
  AND
          “DateLessThan”: {“awsCurrentTime”: “2016-02-01”},
  AND
          “IpAddress”: { “awsSourceIp”: [
            “10.0.1.0/16”, “192.168.1.0/16”
          ]}
                          OR
        }
```

POLICY VARIABLES

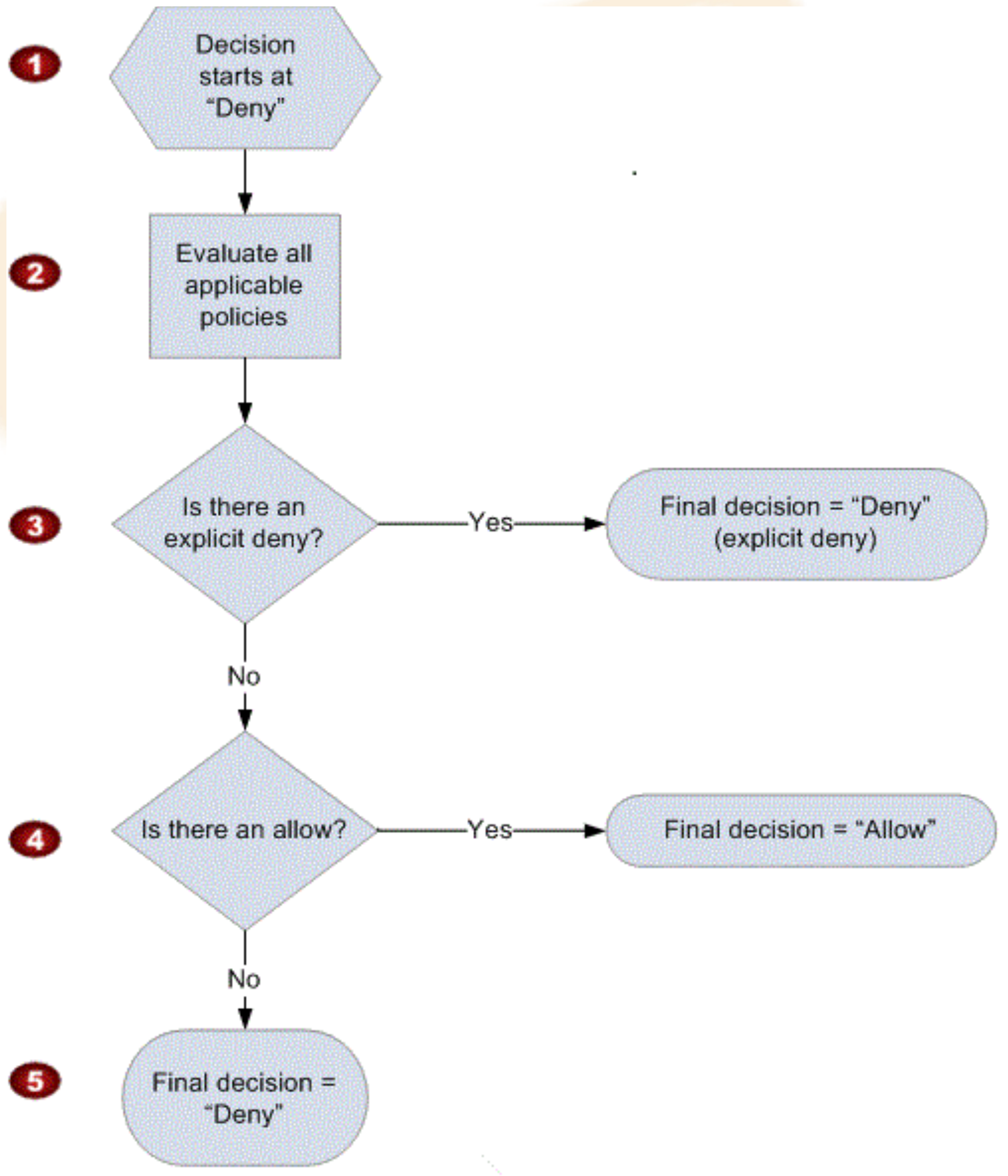
- Like environment variables in Bash
  - aws:SourceIP, aws:CurrentTime, aws:username, aws:userid, aws:principaltype
  - graph.facebook.com:id, www.amazon.com:user_id, ec2:SourceInstanceARN

```
{
  “Version”: “2012-10-17”,
  “Statement”: [{
    “Effect”: “Allow”,
    “Action”: “s3:*”,
    “Resource”: [
       “arn:aws:s3::bucket/${aws:username}”,
       “arn:aws:s3::bucket/${aws:username}/*”,
  }]
}
```

Must have version if the policy has policy variables, and must be “2012-10-17”

POLICY EVALUATION RULES

- Root account is not limited by policy
- Two simple rules
  1. Default is "Deny" (except root account)
  2. "Deny" always overrides "Allow"


1
Decision starts at "Deny"
2
Evaluate all applicable policies
3
Is there an explicit deny?
Yes
Final decision = "Deny" (explicit deny)
No
4
Is there an allow?
Yes
Final decision = "Allow"
No
5
Final decision = "Deny"

POLICY EVALUATION EXAMPLES

```
{
  “Statement”: [{
    “Effect”: “Allow”,
    “Action”: “s3:DeleteBucket”,
    “Resource”: “s3"
  }]
}
```

```
{
  “Statement”: [{
    “Effect”: “Allow”,
    “Action”: “*”,
    “Resource”: “s3"
  },
  {
    “Effect”: “Deny”,
    “Action”: “s3:DeleteBucket”,
    “Resource”: “s3"
  }]
}
```

```
{
  “Statement”: [{
    “Effect”: “Allow”,
    “Action”: “s3:DeleteBucket”,
    “Resource”: “s3"
  },
  {
    “Effect”: “Deny”,
    “Action”: “s3:DeleteBucket”,
    “Resource”: “s3"
  }]
}
```

MANAGED AND INLINE POLICIES

- Managed Policies
  - Attachable to users, groups, roles, NO RESOURCES
  - AWS Managed Policies (Predefined policies for common cases)
  - Customer managed policies
    - 5K size limit
    - 5 version in case of error
    - limit to 10 policy for each resource
- Inline Policies
  - Embedded in users, group, role, or RESOURCES
  - 2K per user, 5K per group, 10K per role

ROLE AND INSTANCE PROFILE

- Role
  - An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user.
  - You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.
- Instance profile
  - Amazon EC2 uses an instance profile as a container for an IAM role. When you create an IAM role using the console, the console creates an instance profile automatically and gives it the same name as the role it corresponds to. If you use the AWS CLI, API, or an AWS SDK to create a role, you create the role and instance profile as separate actions, and you might give them different names.

EXAMPLES

- Policy Editor
- Policy Simulator
- Allow Users to Access a Personal "Home Directory" in Amazon S3
- Deny Users to Launch Certain Types of EC2 Instances
- Allow User to Start, Stop, and Terminate his/her own instances
- Decode the EC2 authorization message

PACKER

PACKER

- An AMI build tool for multiple platforms
  - Automation, version tracking
  - Template
  - Supported platforms: https://www.packer.io/intro/platforms.html
- Examples
  - Build a customized Amazon Linux AMI

“

Whenever you find yourself on the side of majority, it is time to pause and reflect.

-Mark Twain

CLOUDFORMATION

WHY NOT CLOUDFORMATION

- Long, long JSON, error-prone
  - Parenthesis matching
  - Indents
  - Commas
- Now a little bit better: YAML
  - Still not for human…
- Not programmable
- Don't believe me?
  - https://s3-us-west-2.amazonaws.com/cloudformation-templates-us-west-2/EC2InstanceWithSecurityGroupSample.template

TERRAFORM

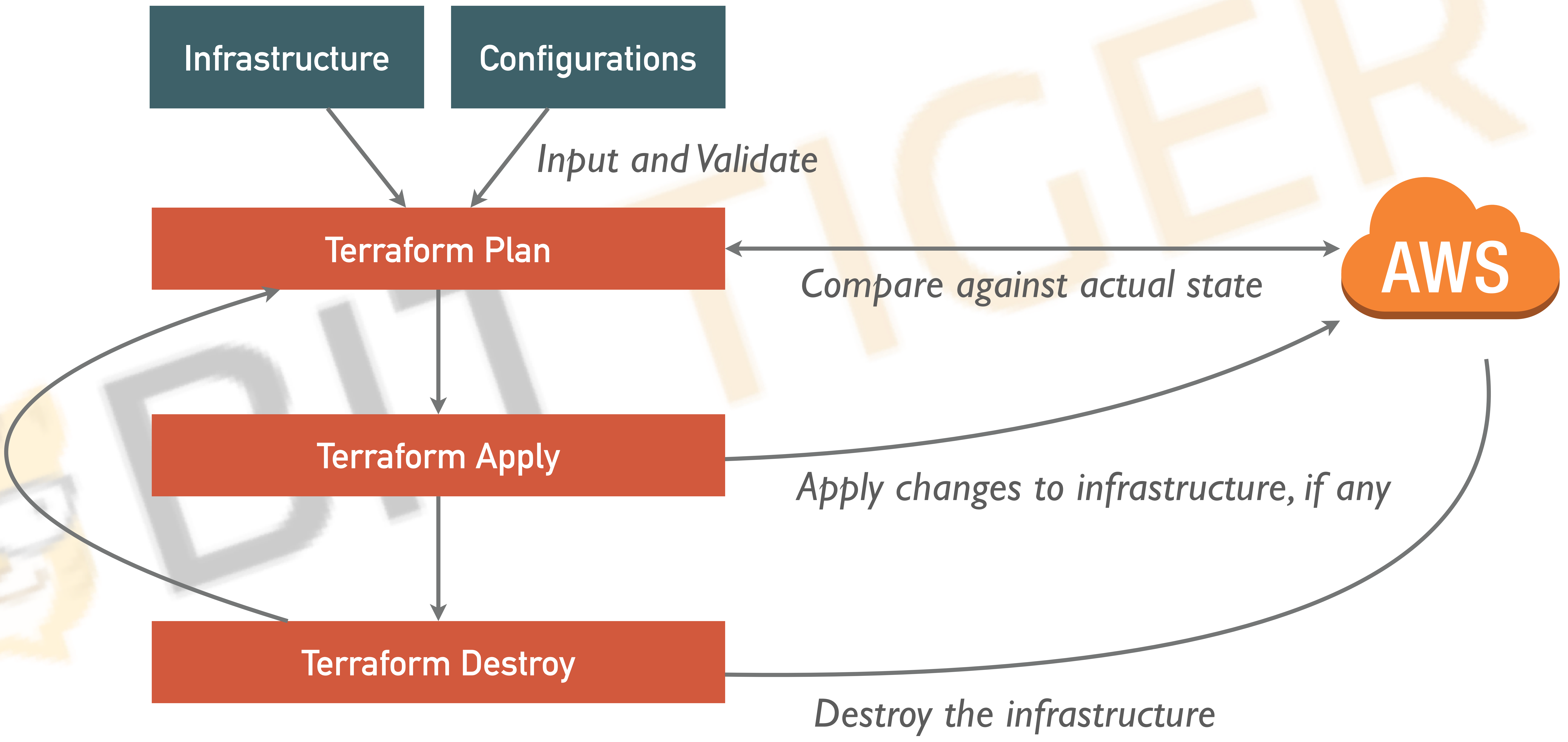
WHY TERRAFORM

- Code one, run on multiple platforms
  - AWS, Azure, Google Cloud Platform, AliCloud…
- Much, much cleaner and easier than CloudFormation
  - Using Modules
- Programmable!
- Configurable

TERRAFORM COMPONENTS

- Provider
- Resource
- Configurations (variables)
- Command lines
  - plan
  - apply
  - destroy

TERRAFORM WORKFLOW



PROVIDER

- Abstraction of a Cloud provider

```
provider "aws" {
    access_key = "${var.aws_access_key}"
    secret_key = “${var.aws_secret_key}"
    profile = “${var.aws_profile}”
    region = "us-east-1"
}
```

RESOURCES

- Abstraction of a cloud resource

```
resource "aws_instance" "web" {
    ami = “${var.ami_id}”
    instance_type = "t2.micro"
    tags {
        Name = "Demo"
    }
}
```

```
resource "aws_ebs_volume" "example" {
    availability_zone = "us-west-2a"
    size = 40
    tags {
        Name = "Demo"
    }
}
```

```
resource "aws_subnet" "subnet" {
  vpc_id = "${var.security_group_id}"
  cidr_block = "10.0.1.0/24"
}
```

CONFIGURATIONS

- Key-value pairs used as input of Terraform

```
variable "key" {
  type = "string"
}

variable "images" {
  type = "map"

  default = {
    us-east-1 = "image-1234"
    us-west-2 = "image-4567"
  }
}

variable "zones" {
  default = [ "us-east-1a", "us-east-1b"]
}
```

STATE

- State
  - A full, detailed, formal description of the infrastructure, (using JSON)
- Real State
  - The actual state of running resources
- Local State
  - Last synchronized state with real state, stored in local disk
- Remote State
  - Last synchronized state with local state, stored in remote storage, such as S3

COMMAND LINE

- Plan
  - Show what will be executed
- Apply
  - Actually execute the plan
- Destroy
  - Destroy the infrastructure
- Output
  - Output pre-defined information and format

DEMO

- Launch a cluster with 1 webserver, 2 mapper, and 1 reducer
- Add one more reducer
- Change the instance type of webserver
- Change the security group

ANSIBLE

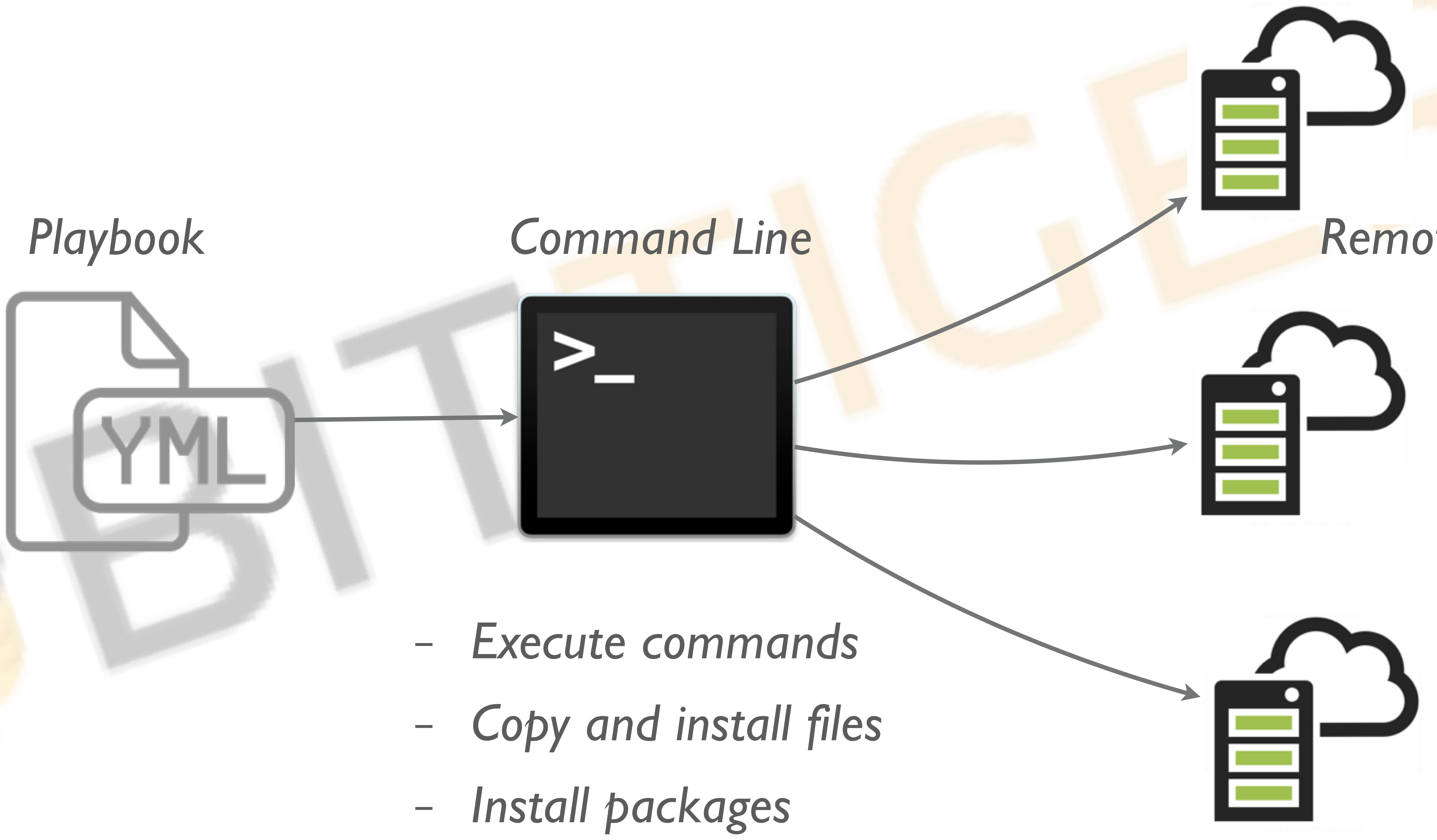
ANSIBLE

- A collective configuration tools
    - ssh only, no pre-installed package required
    - Daemon less
    - Run commands on ad-hoc machines
    - Configure machines based on their properties
    - Idempotency

ANSIBLE CONCEPTS

- Controller machine
- Inventory
- Playbook
- Task
- Module
- Facts
- Handlers

ANSIBLE WORKFLOW



- *Execute commands*
- *Copy and install files*
- *Install packages*

INVENTORY

- A list of machines to be managed by Ansible
  - IP or DNS name
  - Grouped list
  - Groups of groups
- Location
  - /etc/ansible/host
  - export ANSIBLE_HOST=path
  - ansible -i path

```
[webserver]
web.bittiger.info
53.12.34.56

[worker]
worker.bittiger.info

[load_balancer]
load[0:2].us-west-2.bittiger.info
load[0:4].us-east-1.bititger.info

[database]
db.bittiger.info
{{db_backup_server}}

[us-west-2:children]
webserver
worker
database
```

AD-HOC COMMAND

- Execute command on ad-hoc hosts

```
ansible all -i ansible_hosts -m ping
ansible all -m ping
ansible webserver -m ping
ansible db.bittiger -m ping

ansible all -m command -a date
ansible all -a date

ansible all -a reboot
ansible all -a reboot -s
ansible all -a reboot -s -K
```

PLAYBOOK AND TASK

- A collection of plays
- YAML

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=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
```

HANDLERS

- A Handler is exactly the same as a Task (it can do anything a Task can), but it will run when called by another Task.

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed update_cache=true
     notify:
      - Start Nginx

  handlers:
   - name: Start Nginx
     service: name=nginx state=started
```

FACTS

- Automatically collected information of target machines
  - Discovered, not by set
  - Can be used in playbook and templates

```
user www-data www-data;
worker_processes {{ ansible_processor_cores * ansible_processor_count }};
pid /var/run/nginx.pid;
```

MODULES

- Predefined units of work shipped to remote machines (library)
    - Cloud, Clustering, Command, Database, Files, Monitoring, Network, Storage, System, Utility, Web…
    - User-build modules

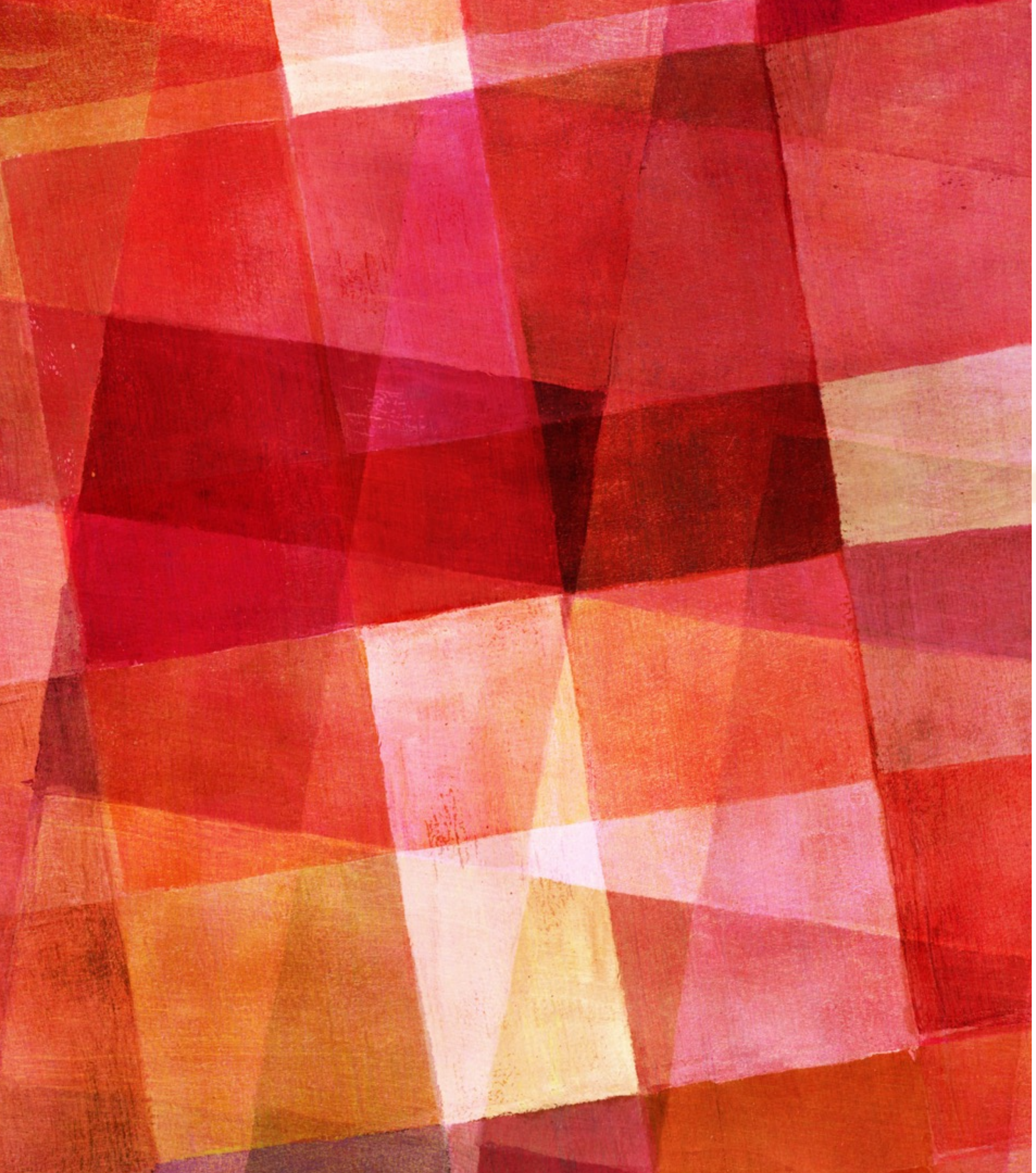
```
ansible -m command -a ls /


- name: return motd to registered var
  command: cat /etc/motd
  register: mymotd


- name: Run the command if the specified file does not exist.
  command: /usr/bin/make_database.sh arg1 arg2 creates=/path/to/database
```

DEMO

- Run commands on all nodes, mapper nodes only
- Run playbook to install apache server on webserver instance
- Run playbook to install spark on mapper nodes
- Get instance facts

HOMEWORK

- Register a domain name in Route53
    - http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/registrar.html
- Learn JSON
    - https://www.sitepoint.com/10-example-json-files/
    - Validator: https://jsonformatter.curiousconcept.com/
- Learn YAML
    - https://learnxinyminutes.com/docs/yaml/
- Create another SSH key for deployment using ssh-keygen
    - “id_bitbucket_deploy”
    - No password
    - Upload public key to bitbucket
    - https://confluence.atlassian.com/bitbucket/add-an-ssh-key-to-an-account-302811853.html



QUESTIONS

- bittiger-aws@googlegroups.com

BITTIGER