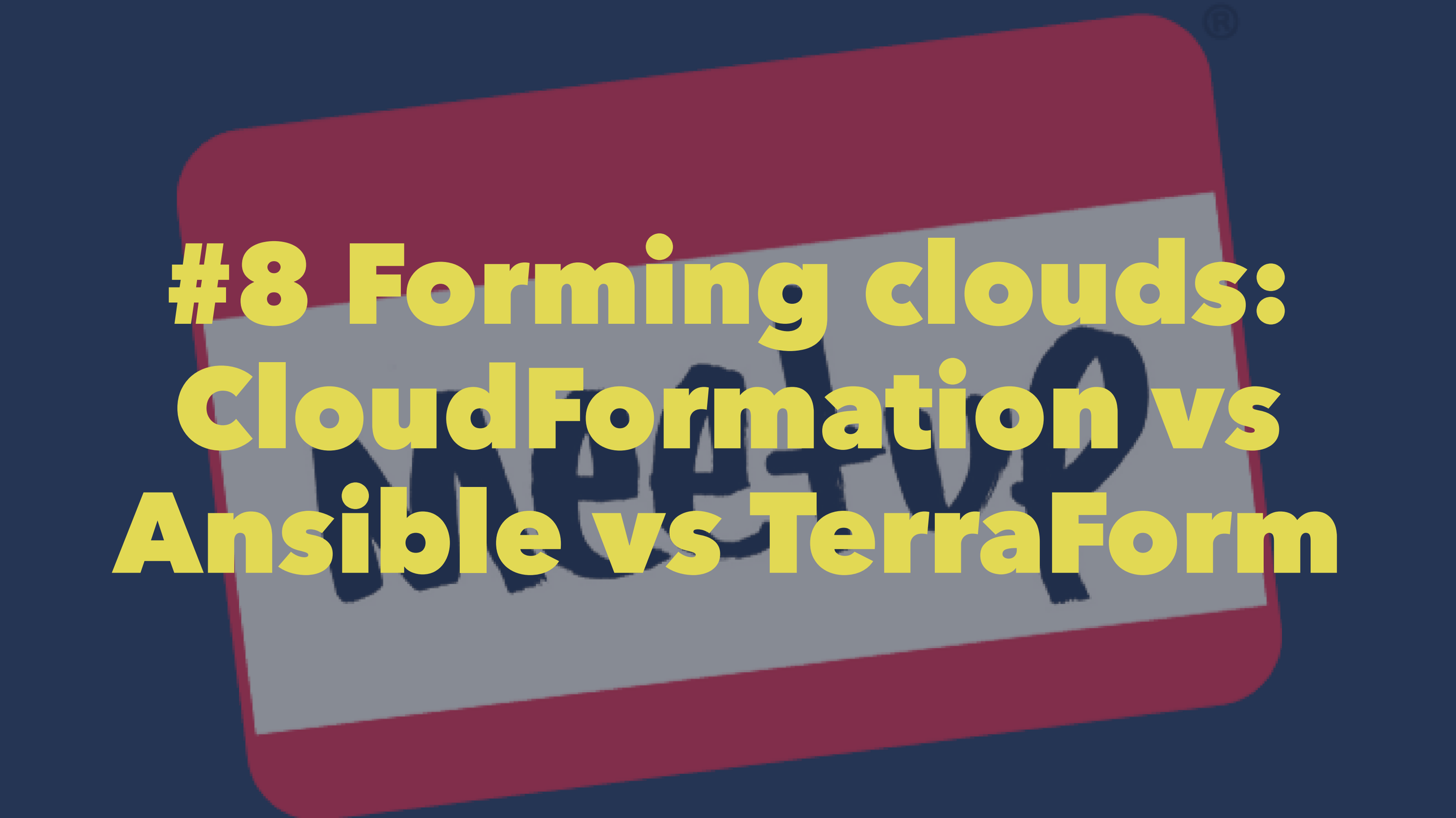

#8 Forming clouds:
CloudFormation vs
Ansible vs TerraForm

skyscrapers

The Cloud & Ops Experts

Geert Theys

Sales guy @skyscrapers

Still likes to get his hands dirty.

github.com/gtheys
twitter.com/toadi
geerttheys.com

Mattias Gees

Cloud Engineer @skyscrapers

Likes to experiment!

github.com/mattiasgees
twitter.com/mattiasgees
blog.mattiasgees.be

CloudFormation <> Terraform <> Ansible for AWS provisioning

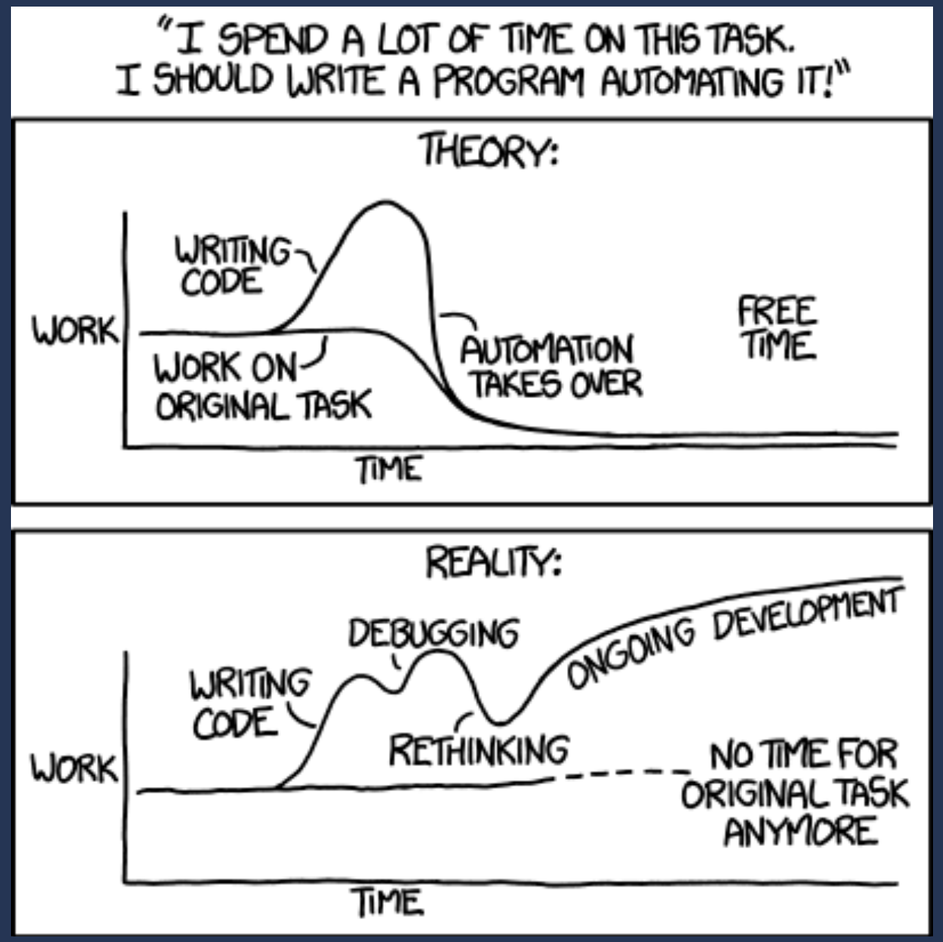
All different tools

and there are many others!

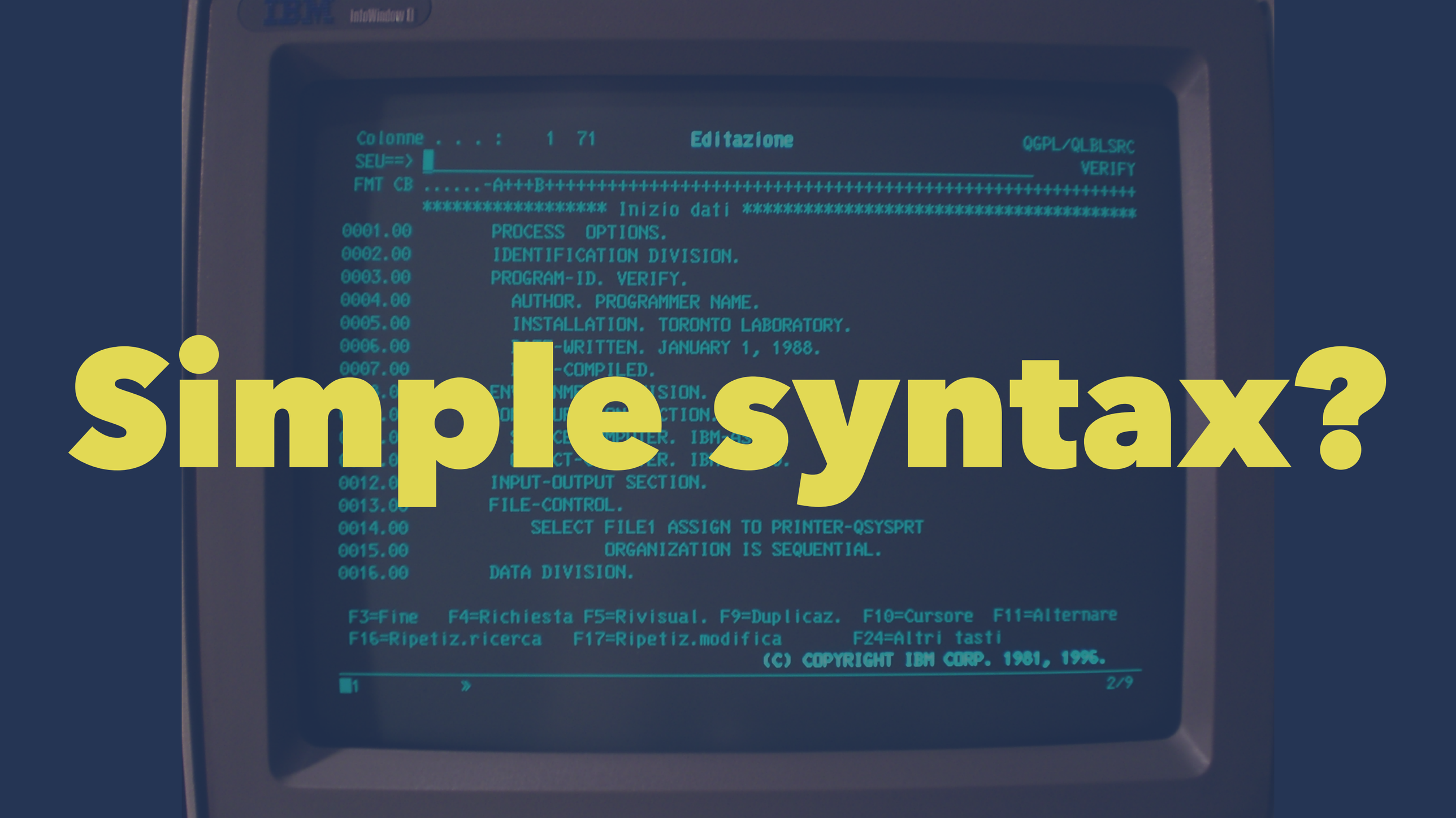
All have their advantages

And disadvantages

Before we start!

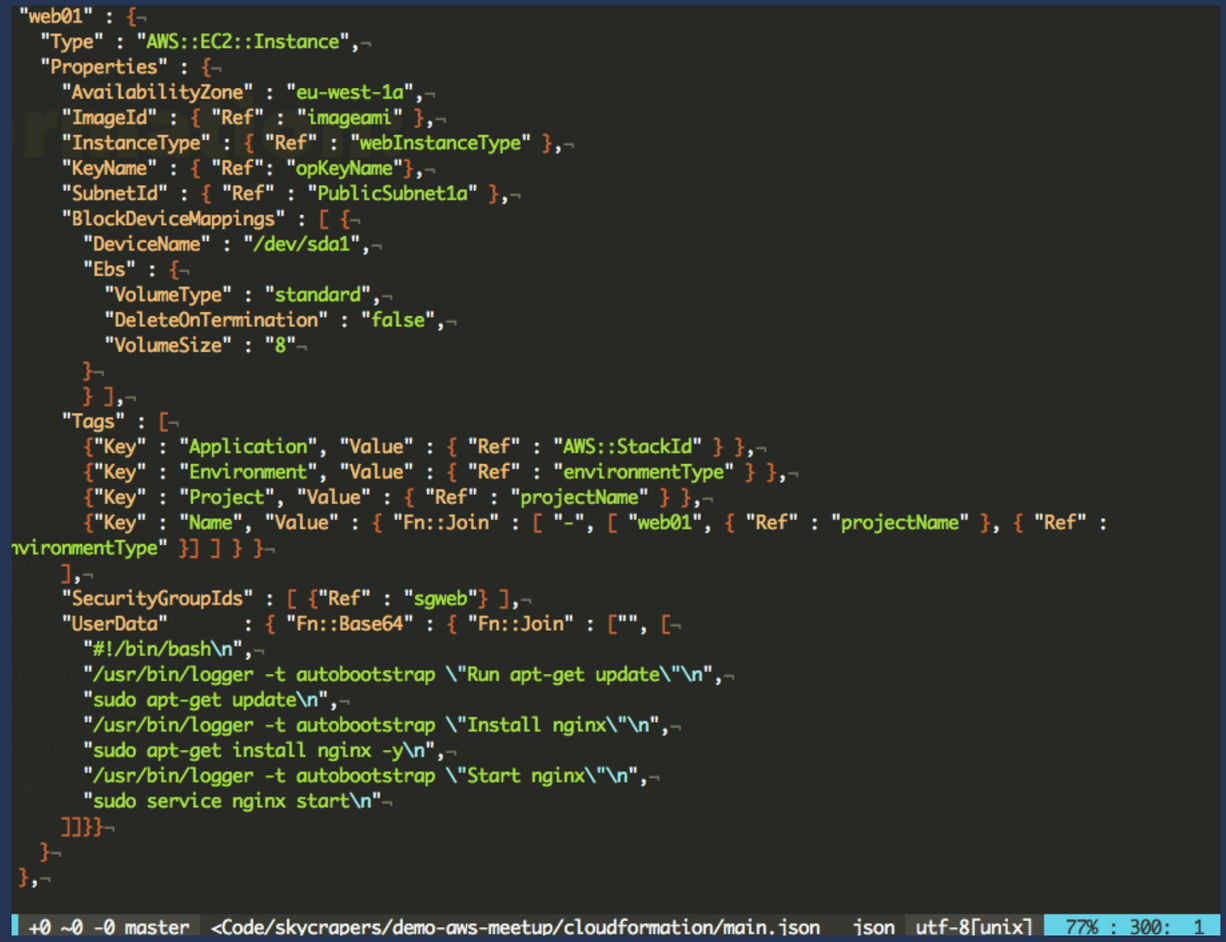


Lets compare



Simple syntax?

CloudFormation:



```
"web01" : {¬
  "Type" : "AWS::EC2::Instance",¬
  "Properties" : {¬
    "AvailabilityZone" : "eu-west-1a",¬
    "ImageId" : { "Ref" : "imageami" },¬
    "InstanceType" : { "Ref" : "webInstanceType" },¬
    "KeyName" : { "Ref": "opKeyName"},¬
    "SubnetId" : { "Ref" : "PublicSubnet1a" },¬
    "BlockDeviceMappings" : [ {¬
      "DeviceName" : "/dev/sda1",¬
      "Ebs" : {¬
        "VolumeType" : "standard",¬
        "DeleteOnTermination" : "false",¬
        "VolumeSize" : "8"¬
      }¬
      } ],¬
    "Tags" : [¬
      {"Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },¬
      {"Key" : "Environment", "Value" : { "Ref" : "environmentType" } },¬
      {"Key" : "Project", "Value" : { "Ref" : "projectName" } },¬
      {"Key" : "Name", "Value" : { "Fn::Join" : [ "-", [ "web01", { "Ref" : "projectName" }, { "Ref" :
nvironmentType" }] ] } }¬
    ],¬
    "SecurityGroupIds" : [ {"Ref" : "sgweb"} ],¬
    "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [¬
      "#!/bin/bash\n",¬
      "/usr/bin/logger -t autobootstrap \"Run apt-get update\"\n",¬
      "sudo apt-get update\n",¬
      "/usr/bin/logger -t autobootstrap \"Install nginx\"\n",¬
      "sudo apt-get install nginx -y\n",¬
      "/usr/bin/logger -t autobootstrap \"Start nginx\"\n",¬
      "sudo service nginx start\n"¬
    ]]}}¬
  }¬
},¬
+0 ~0 -0 master  <Code/skycrapers/demo-aws-meetup/cloudformation/main.json   json  utf-8[unix]  77% : 300:  1
```

Terraform:

```
resource "aws_instance" "web" {
  count = "${var.web_nodes}"
  ami = "${var.ami}"
  instance_type = "${var.instance_type}"
  subnet_id = "${element(aws_subnet.public_subnets.*.id, count.index)}"
  key_name  = "${var.key_name}"
  security_groups  = ["${aws_security_group.sg_web.id}"]
  user_data = "${template_file.metadata_web.rendered}"

  root_block_device {
    volume_type = "standard"
    volume_size = "8"
    delete_on_termination = "false"
  }

  tags {
    Name = "${var.project}-${var.environment}-web0${count.index + 1}"
    Environment = "${var.environment}"
    Project = "${var.project}"
  }
}

resource "template_file" "metadata_web" {
    filename = "templates/metadata.tpl"
}
```

Ansible:

```
- name: WebServer | Create the WebServer Instance(s)¬
  local_action:¬
    module: ec2¬
    region: "{{ vpc_region }}"¬
    group: "{{ web_security_groups[0].sg_name }}"¬
    keypair: "{{ key_name }}"¬
    instance_type: "{{ web_instance_type }}"¬
    image: "{{ imgae_id.ami }}"¬
    vpc_subnet_id: "{{ item }}"¬
    assign_public_ip: True¬
    wait: True¬
    wait_timeout: 600¬
    user_data: |¬
              #!/bin/sh¬
              sudo apt-get install nginx -y¬
    instance_tags:¬
      Name: "{{ vpc_name }}_WEB_Instance"¬
      Environment: "{{ ENV }}"¬
      Role: "{{ server_role }}"¬
      Application: "{{ application }}"¬
  with_items:¬
     - "{{ public_subnet_1 }}"¬
     - "{{ public_subnet_2 }}"¬
  register: web¬
¬
- name: WebServer | Set the Instances facts¬
  set_fact:¬
    instance_public_ip_1: "{{ web.results[0].instances[0].public_ip }}"¬
    instance_id_1: "{{ web.results[0].instances[0].id }}"¬
    instance_public_ip_2: "{{ web.results[1].instances[0].public_ip }}"¬
    instance_id_2: "{{ web.results[1].instances[0].id }}"¬
¬
# You can also choose to use dynamic groups using ec2.py¬
 - name: WebServer | Add the newly created EC2 instance(s) to the local host group (located inside the
directory)¬
YAML  +0 ~0 -0 master  <p/ansible/tasks/webserver.yml   yaml  utf-8[unix]  38% :  37:  1  ! trailing[70]
```

Which one was the easiest to read?

Easy to install?

Package managers:

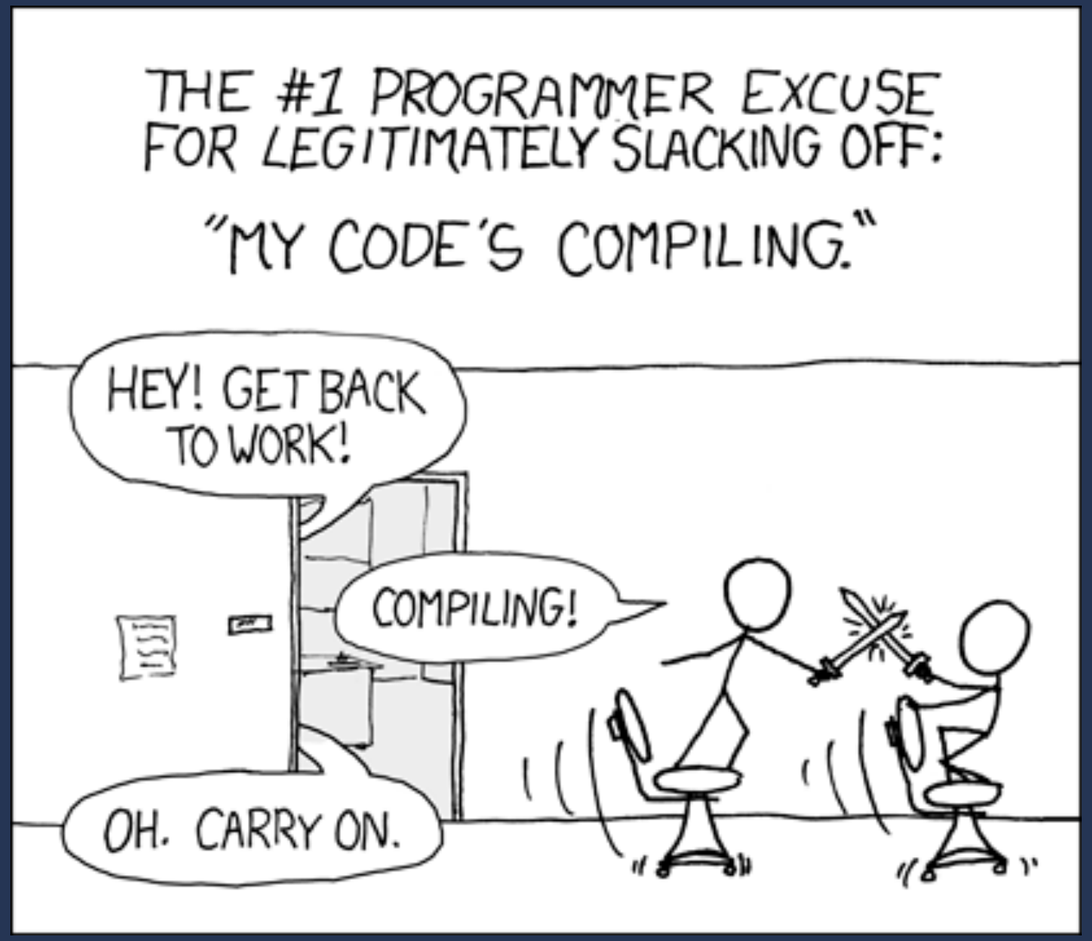
- `brew install`
- `apt-get install`
- `yum install`

Warning: None are up to date!

Latest versions pip and/or sources


Safe to use?
Terraform and Ansible have a plan or --check mode

Productive?


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

It's running ;)

- CloudFormation:
  - Parallelizes as much as possible
- Terraform:
  - Use dependency graph and parallelizes as much as possible
  - Partial refresh before changes
- Ansible:

How do they keep state?

- Cloudformation on AWS
- Terraform creates a state file
- Ansible ad hoc state

Do I feel safe

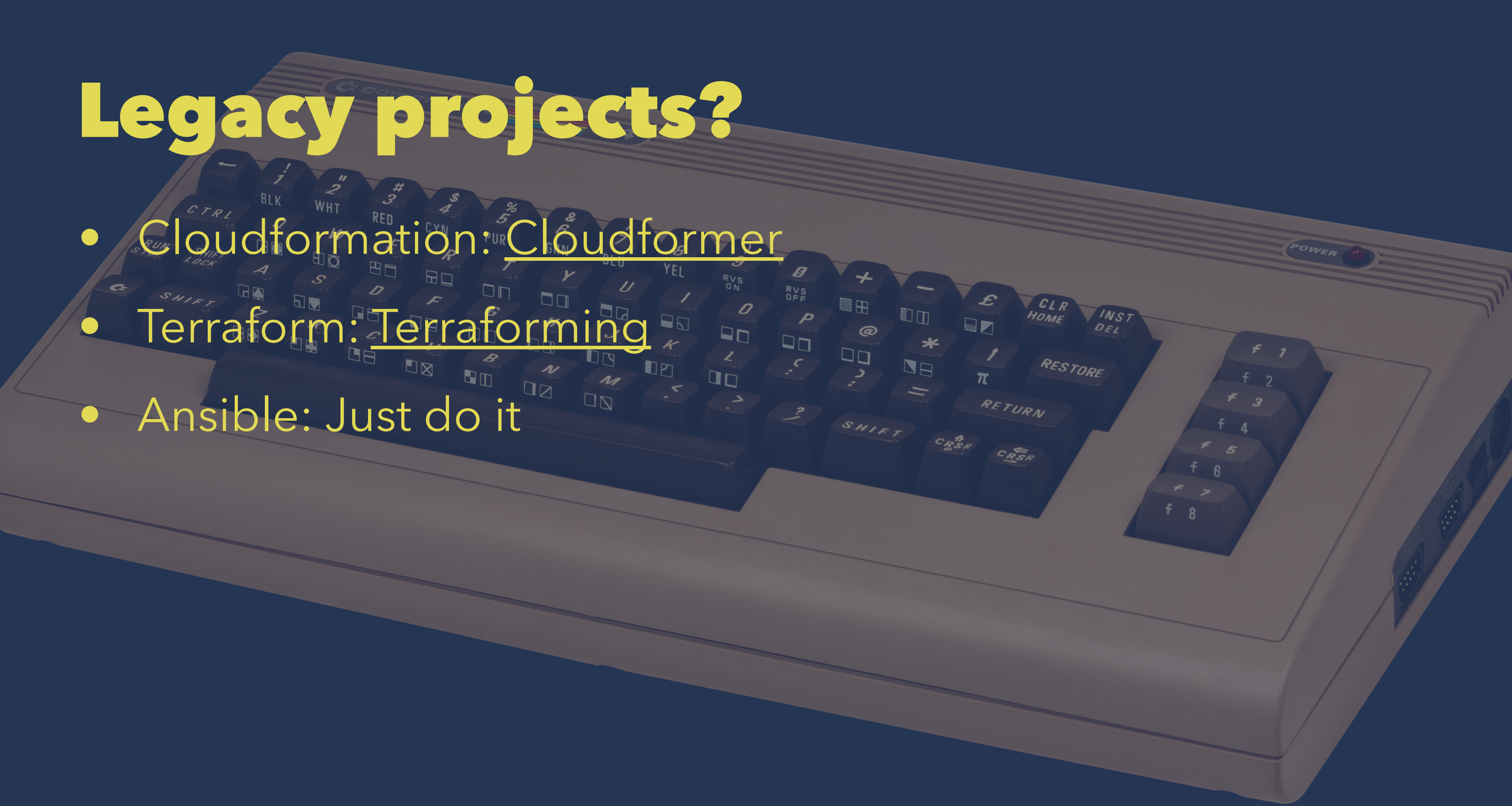
- CloudFormation:
  - Start to pray when you run it
  - Roll back on fail
- Terraform:
  - Partial State gets stored on error (eg. sg gets created not the rules, next run will fix this)
  - Create before destroy

Issues?

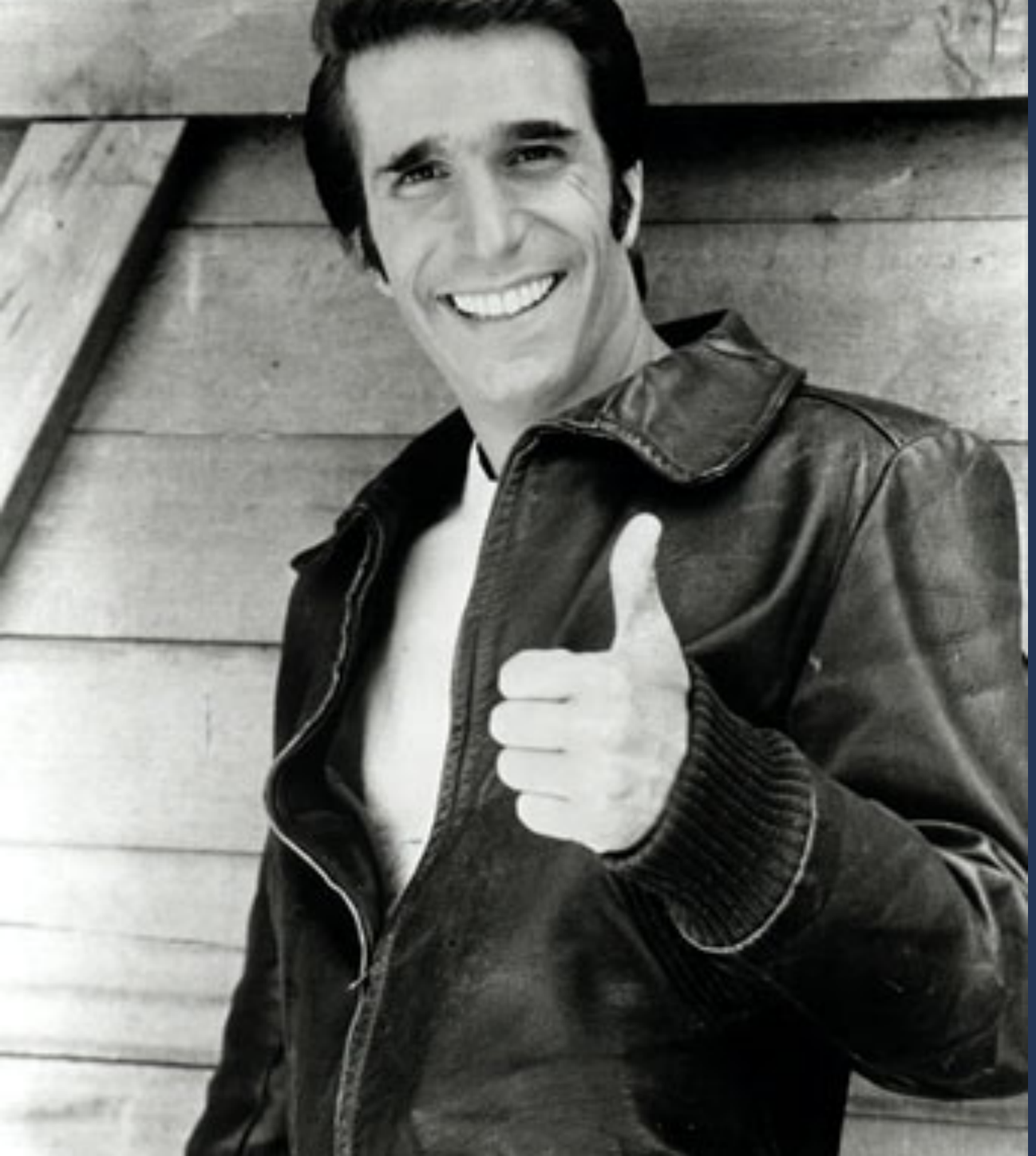
- CloudFormation:
  - JSON
  - No partial run possible
- Terraform:
  - No full coverage of AWS
- Ansible:
  - Not every aws module has --dry-run check mode!

Legacy projects?

- Cloudformation: Cloudformer
- Terraform: Terraforming
- Ansible: Just do it




caps lock
A
Z
X
support
alt
option



Do you look cool when using it?

Do I want to use it?

- Terraform: Destroy ordering, clean and readable, focus on 1 thing
- Ansible: It's simple
- Cloudformation: Who likes JSON?

Do we recommend 1?

Questions?

Presentation + demo on github.com/skyscrapers/demo-aws-meetup