Terraform and the AWS cloud

A Sonic Drive-In Use Case

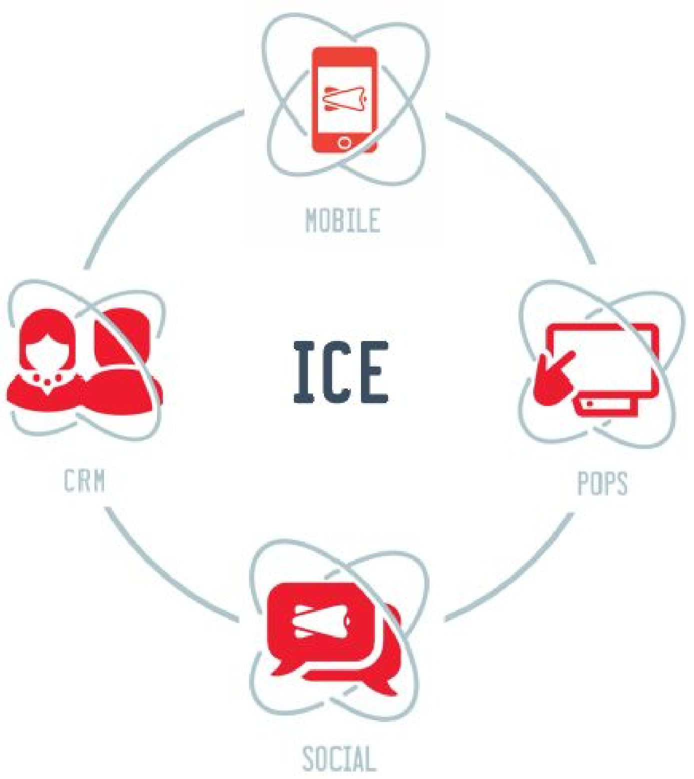
John Wassilak
Caleb Sabatini

Become America's most loved brand!

Sonic Drive In *60+ Years of Being America's Drive-In Experts*

Two Guys *Strong Presence on Traditional Marketing Channels*

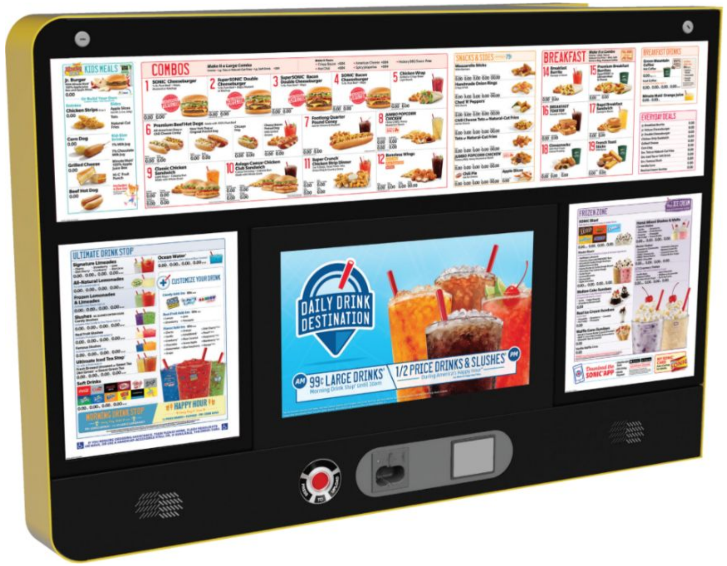
Technology *Integrated Customer Experience*

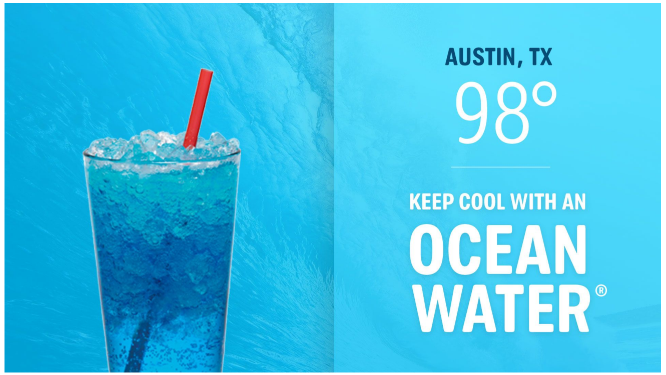


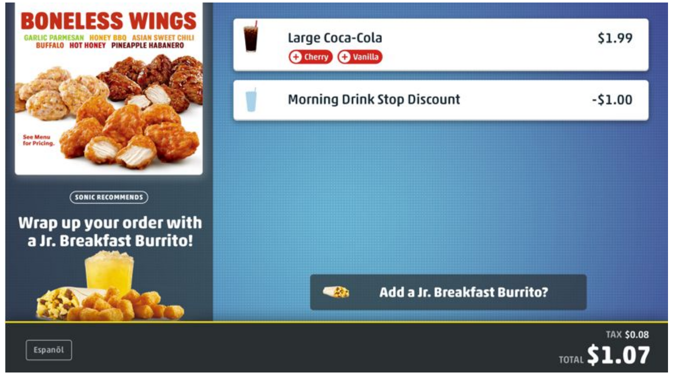
ICE VISION

Deliver the most personalized consumer experience of any Fast Casual or QSR restaurant in America.

POPS *Interacting With The Customer On Lot - Order Confirmation - Upsell*





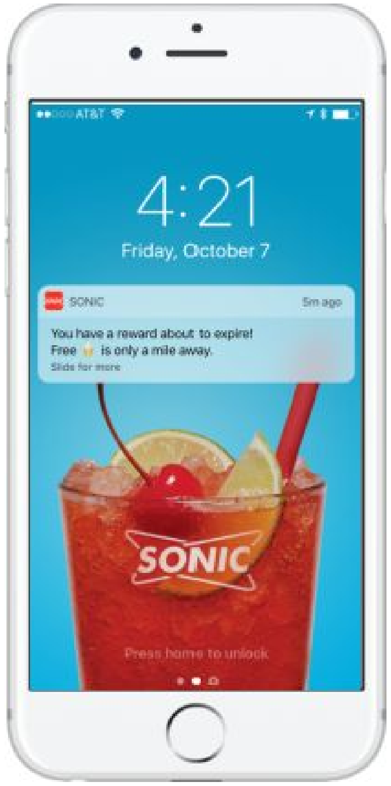


BONELESS WINGS
GARLIC PARMESAN HONEY BBQ ASIAN SWEET CHILI BUFFALO HOT HONEY PINEAPPLE HABANERO
See Menu for Pricing.
SONIC RECOMMENDS
Wrap up your order with a Jr. Breakfast Burrito!
Add a Jr. Breakfast Burrito?
Español

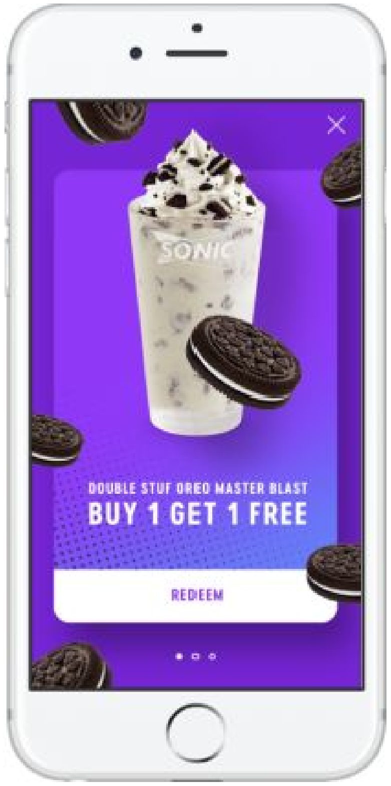
Large Coca-Cola (Cherry, Vanilla)	$1.99
Morning Drink Stop Discount	-$1.00

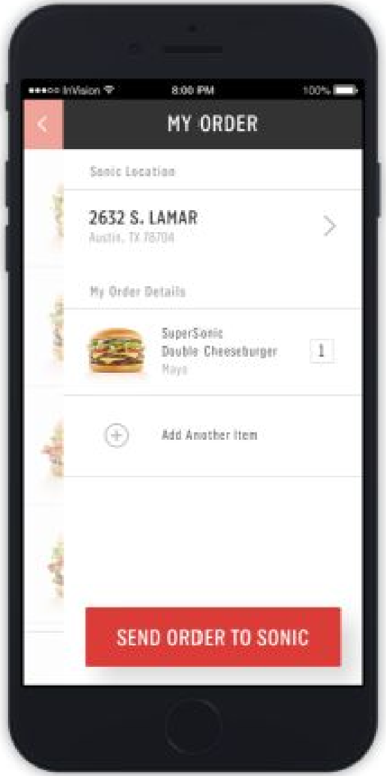
TAX $0.08
TOTAL $1.07

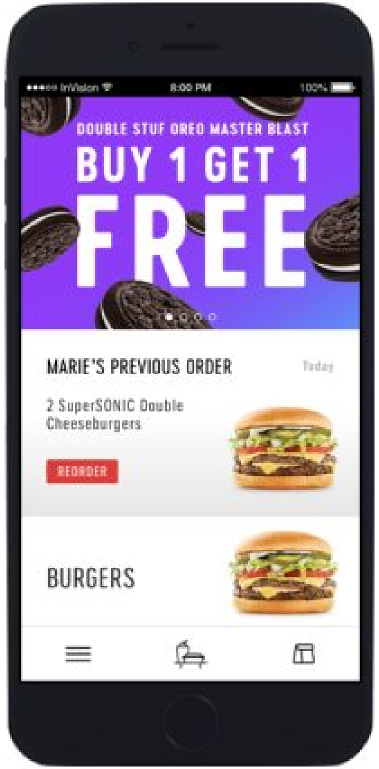
Mobile *Interacting With The Customer Off Lot* - *Notifications - Rewards - (re)Order Ahead*



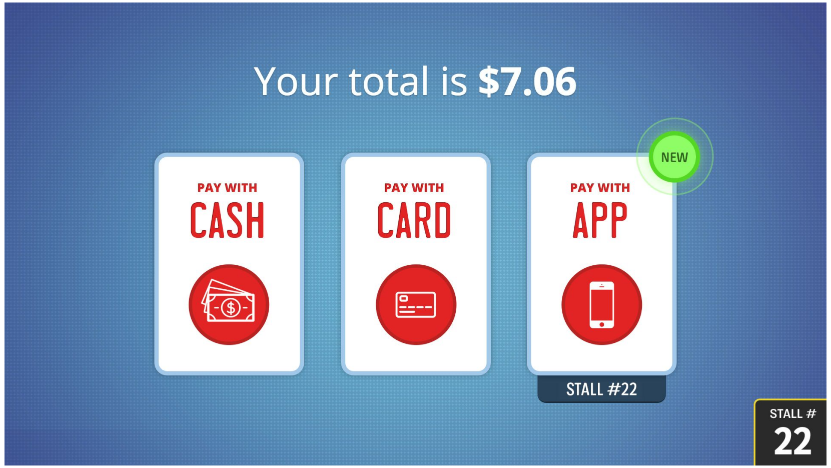
You have a reward about to expire!
Free 🍦 is only a mile away.

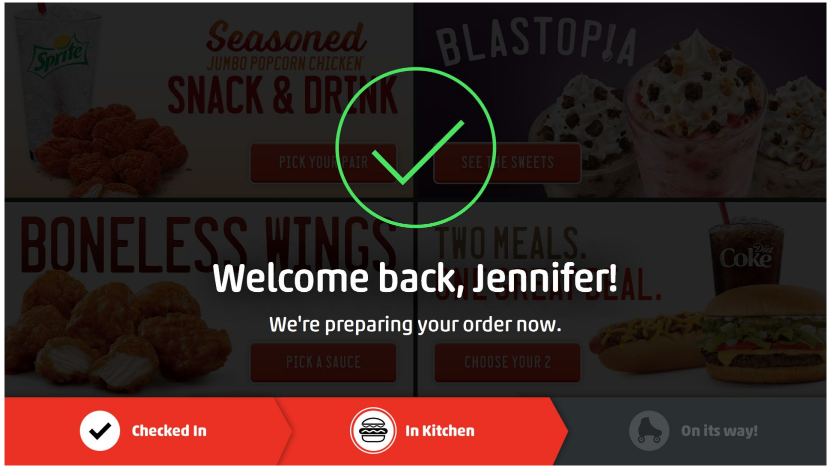






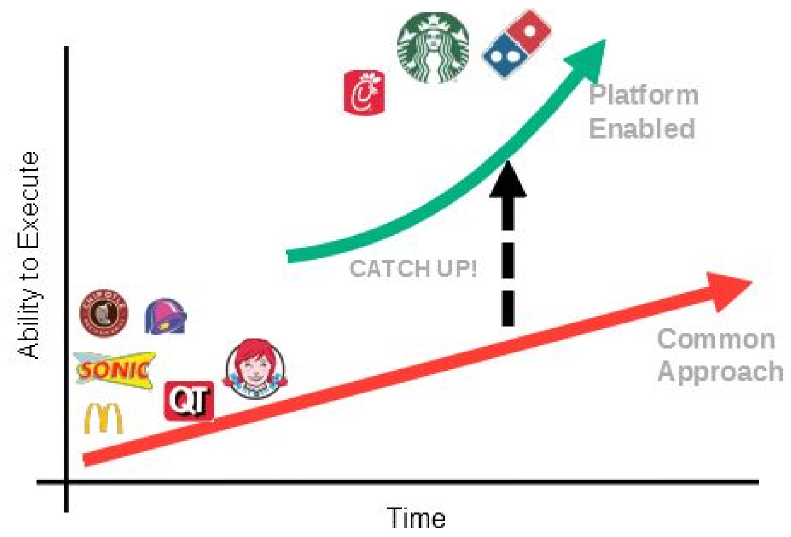
Mobile *Enhancing the On Lot Experience - Mobile Pay - Customized Messaging/Offers*





The Challenge *Sonic is Playing Catch up - Shifting to Platformed Approach*

Compounding effects…



Platform Benefits…

1. Deliver Faster
2. Experiment Responsibly
3. Gain Business Insights
4. Extend Partner Community
5. Consistent Experience

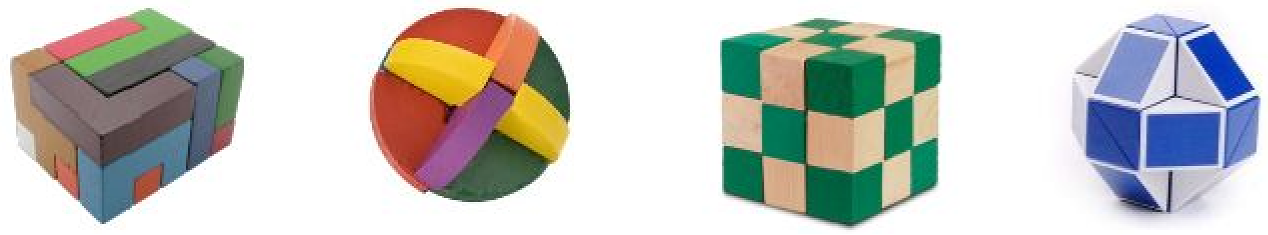
Business results…

1. Faster Time to Market
2. Higher Sales & Profits
3. Optimized IT Costs

The Challenge *Siloed Approach*

MOBILE WEB POPS EMAIL TEXT

Siloed and Non-Integrated!!



Slow to Change

Expensive to Maintain

Difficult to Secure

Hard to Scale

The Challenge *Platformed Approach* - *Speed and Consistency*

MOBILE
WEB
POPS
EMAIL
TEXT
amazon echo

Integrated Digital Platform!!

SONIC
ThoughtWorks
SecureWorks
amazon

Accelerates delivery

Supports Business Change

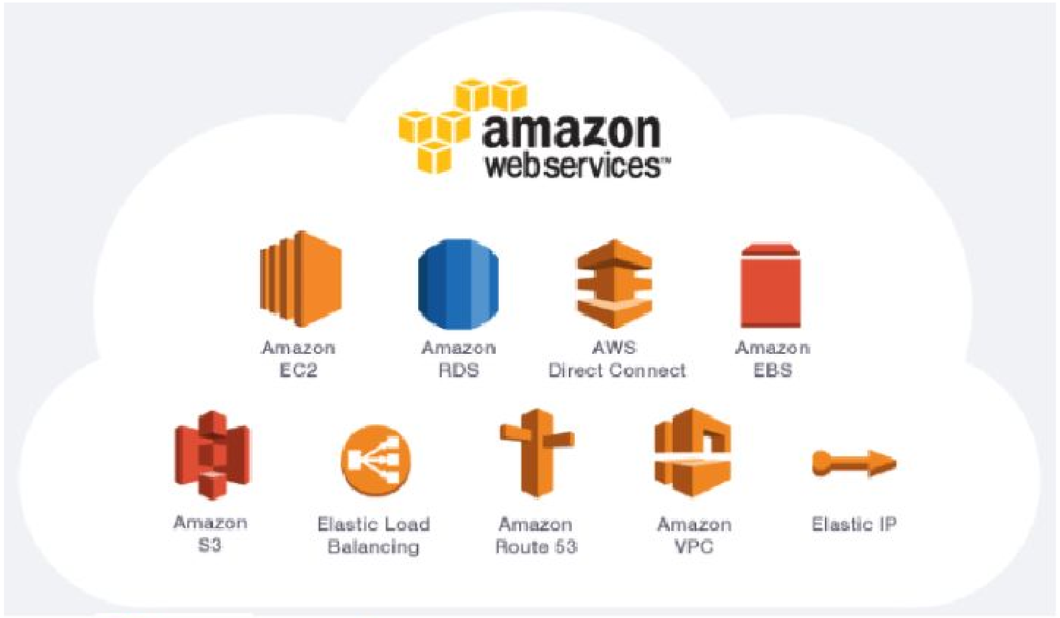
Promotes Security First

Enables Rapid Scalability

Provides an Integrated Customer Experience

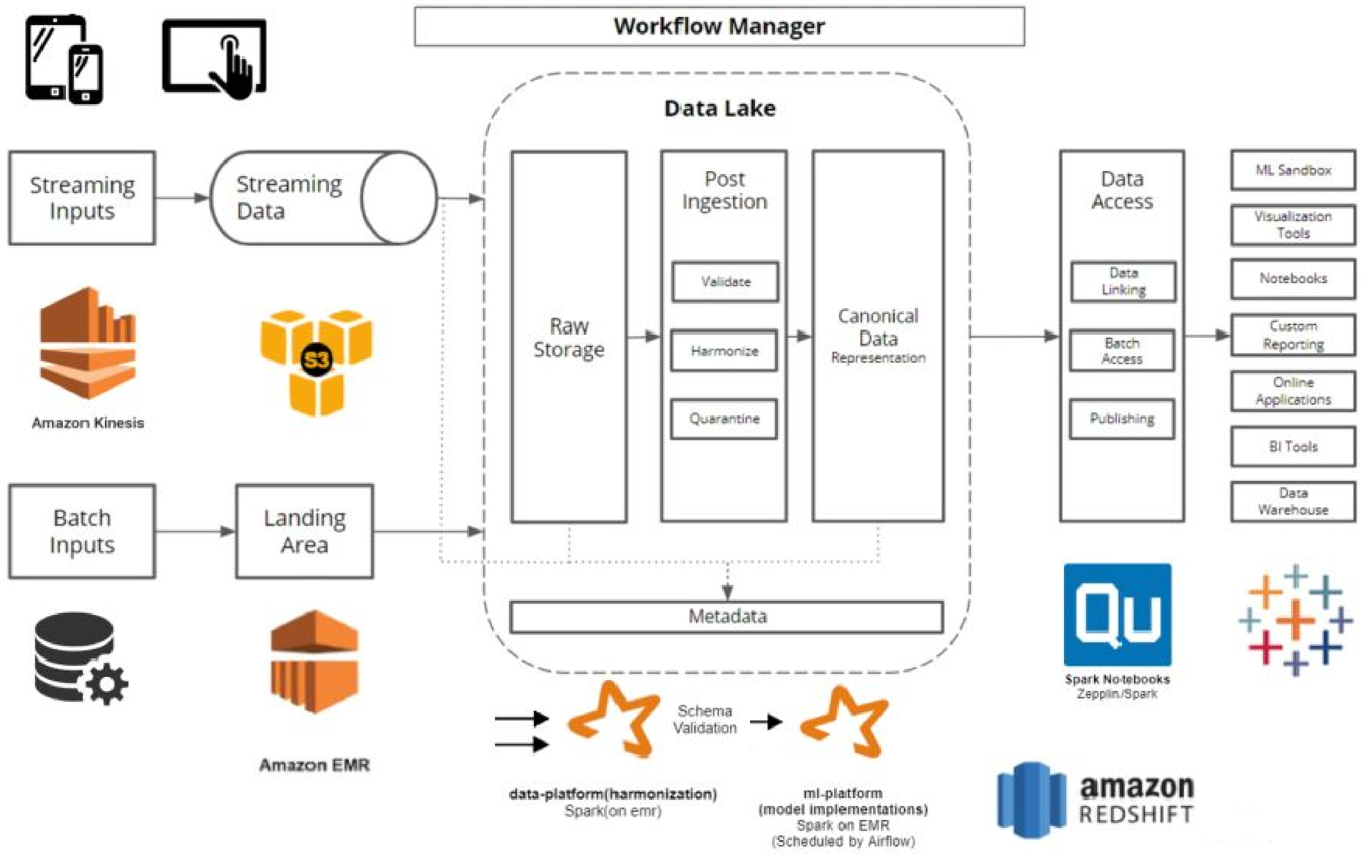
Unlocks Innovation!!

Nuts-n-Bolts *Platform needs to be robust and scalable - Enter AWS and Terraform*



Auth0 spring R/GA appboy {...} SWAGGERhub ThoughtWorks Java CF Pivotal Cloud Foundry go

Back End *(Customer Facing Channels == Large Amounts of Data) == Large Amount of Resources*



Infrastructure Automation *What we need*

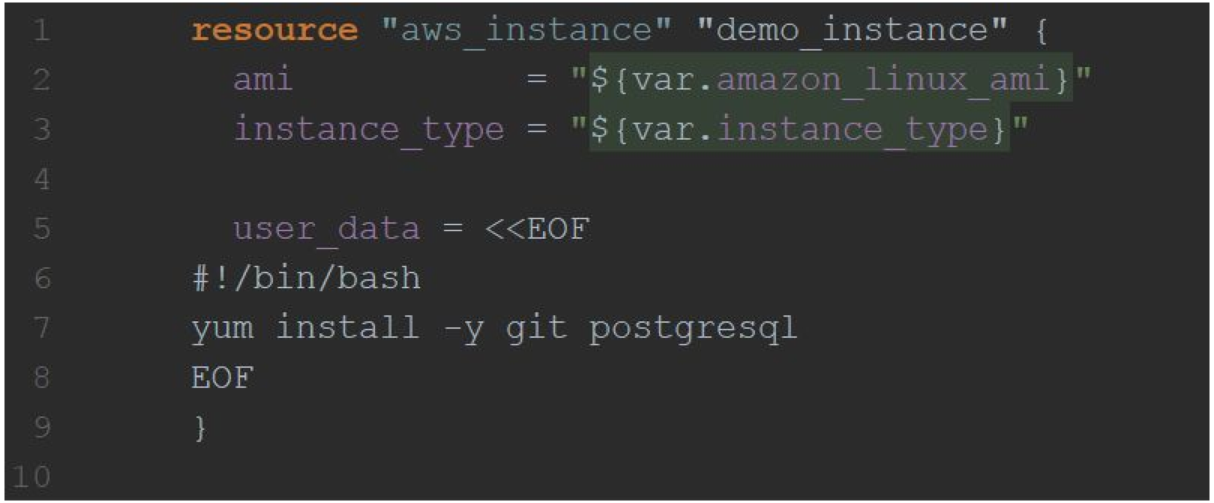
- Infrastructure as Code
    - No clicking around in a web console to spin up resources
    - Reduced risk of mistakes (removing human element)
- Concise Syntax (sorry CloudFormation)
    - Speed of development
    - Maintainability
- Fine grained control of Cloud Resources
    - Identity/Access locked down to least privilege
    - Reducing cost by right-sizing resources.
- Support for more than just AWS

Terraform *What is it? Why we use it*

- Infrastructure management: version, share, and re-use it like your other code
- Cloud agnostic: use one or more cloud providers
    - Amazon Web Services
    - Google Cloud Platform
    - Microsoft Azure
    - OpenStack
- Supports separate plan and apply steps to preview changes
- Efficiency: parallel creation of resources

Terraform *What is it NOT?*

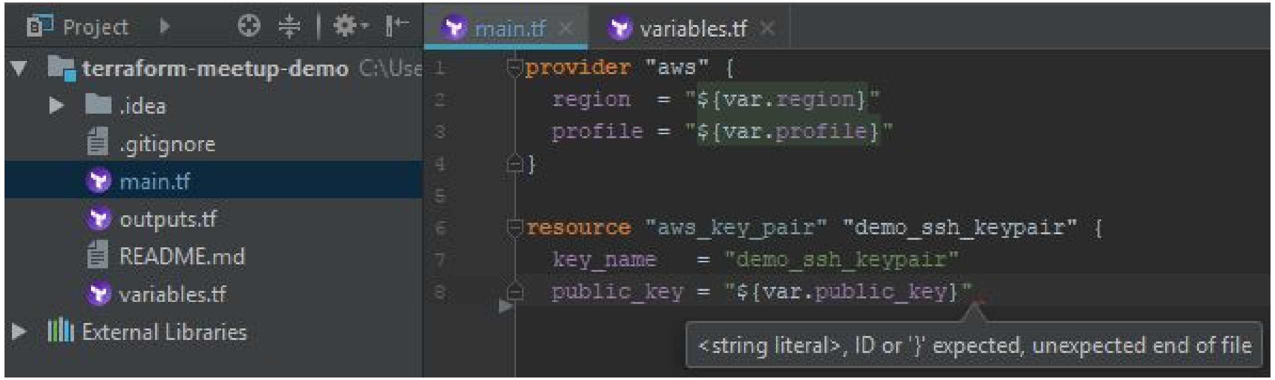
- Doesn’t manage software on machines after create time
- But Terraform can install configuration agents (Chef, Puppet)
- For initial bootstrapping on AWS, we use AMIs or a user_data script



```
resource "aws_instance" "demo_instance" {
  ami           = "${var.amazon_linux_ami}"
  instance_type = "${var.instance_type}"

  user_data = <<EOF
#!/bin/bash
yum install -y git postgresql
EOF
}

```

Using Terraform *Basic Usage*

- Scripts written in HCL - HashiCorp Configuration Language
  - One syntax for multiple cloud providers
  - IDE plugin (JetBrains IntelliJ) provides syntax error checking and highlighting



```
provider "aws" {
  region  = "${var.region}"
  profile = "${var.profile}"
}

resource "aws_key_pair" "demo_ssh_keypair" {
  key_name   = "demo_ssh_keypair"
  public_key = "${var.public_key}"
```

```
<string literal>, ID or '}' expected, unexpected end of file
```

- Run Terraform CLI in the directory with your scripts

Demo

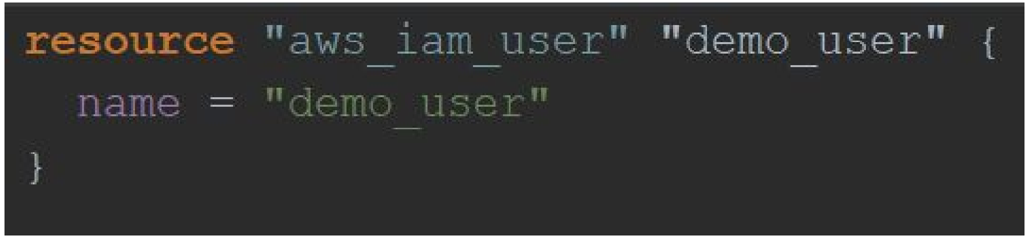
Basic Usage of Terraform

- Plan command
- Apply command
- Console outputs

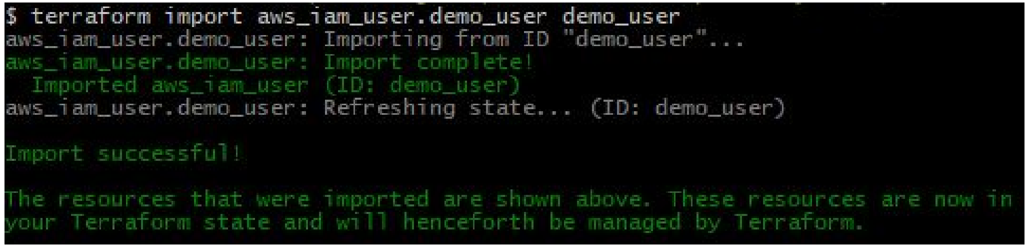
Advanced Terraforming at Sonic

Advanced Usage *Importing existing resources*

- Import command brings an existing resource into Terraform management
- Good for migrating to Infrastructure as Code
- Example: import existing users to add fine-grained access controls



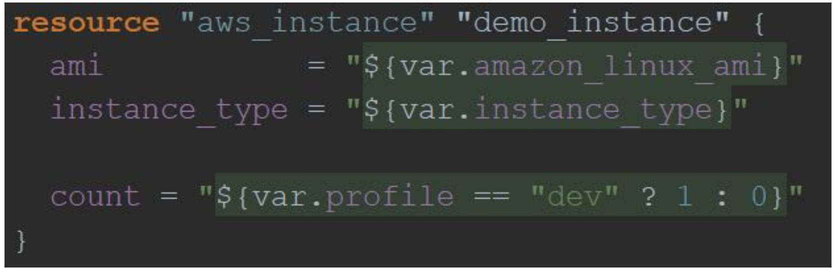
```
resource "aws_iam_user" "demo_user" {
  name = "demo_user"
}
```



```
$ terraform import aws_iam_user.demo_user demo_user
aws_iam_user.demo_user: Importing from ID "demo_user"...
aws_iam_user.demo_user: Import complete!
  Imported aws_iam_user (ID: demo_user)
aws_iam_user.demo_user: Refreshing state... (ID: demo_user)

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

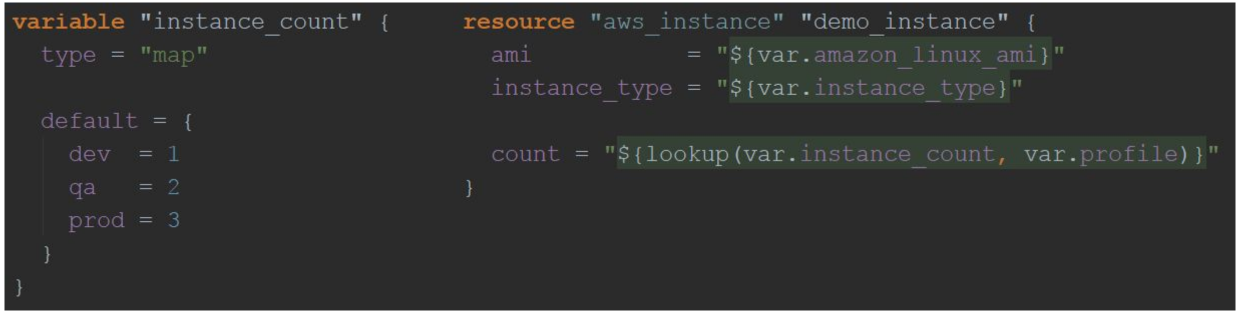
Advanced Usage *Conditionals, Functions*

- Achieve dynamic configurations with If/else conditions



```
resource "aws_instance" "demo_instance" {
  ami           = "${var.amazon_linux_ami}"
  instance_type = "${var.instance_type}"

  count = "${var.profile == "dev" ? 1 : 0}"
}
```

- With more possible values, use the lookup function on a map

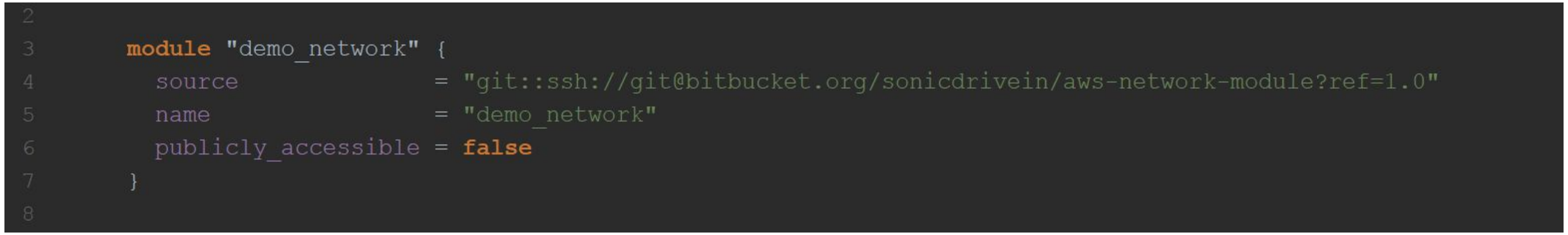


```
variable "instance_count" {
  type = "map"

  default = {
    dev  = 1
    qa   = 2
    prod = 3
  }
}
```

```
resource "aws_instance" "demo_instance" {
  ami           = "${var.amazon_linux_ami}"
  instance_type = "${var.instance_type}"

  count = "${lookup(var.instance_count, var.profile)}"
}
```

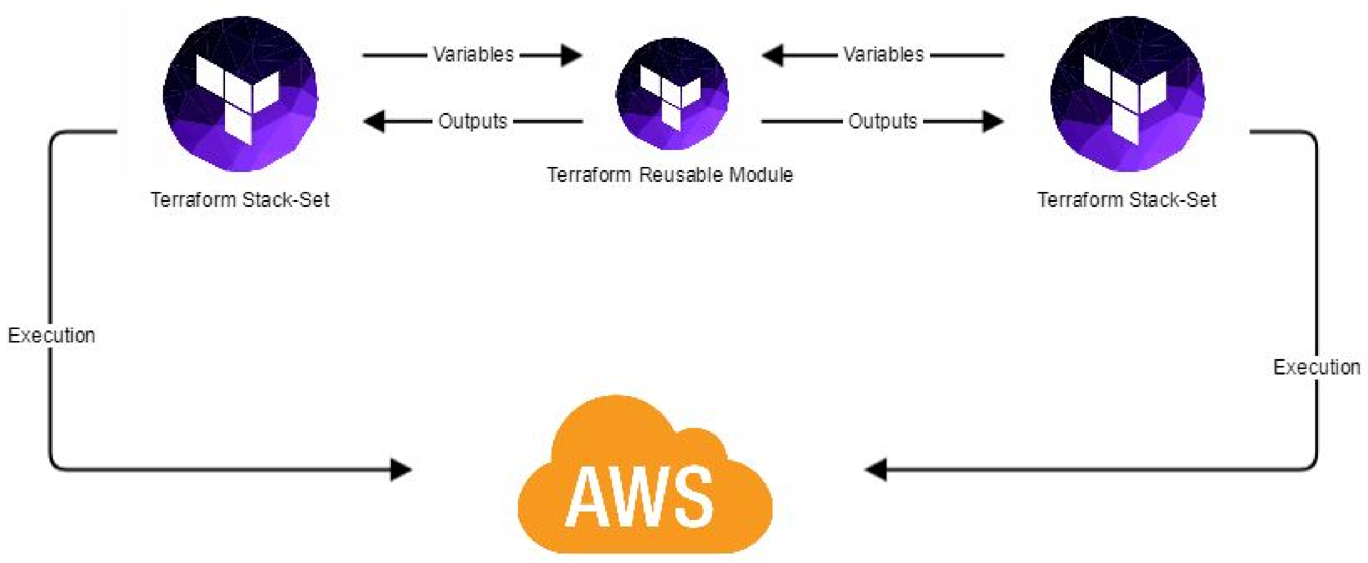
Advanced Usage *Modules*

- Packaged configurations that accept inputs, and create a set of resources
- Enables re-usability for commonly grouped resources
- Source code for modules
  - Local filesystem
  - Remote git repository (Github, Bitbucket)



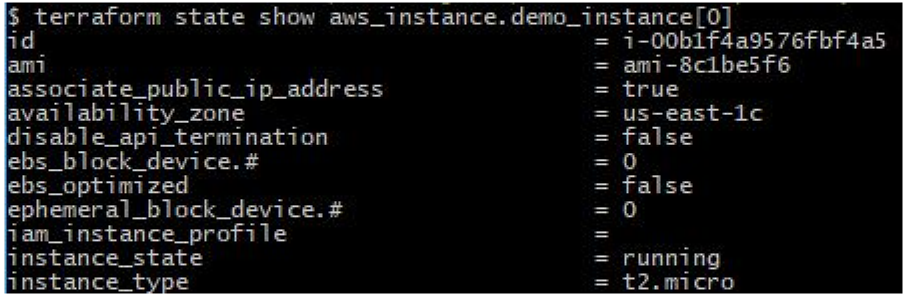
```
module "demo_network" {
  source              = "git::ssh://git@bitbucket.org/sonicdrivein/aws-network-module?ref=1.0"
  name                = "demo_network"
  publicly_accessible = false
}
```

Advanced Usage *Modules*



Lessons Learned *State* - *Local vs Remote*

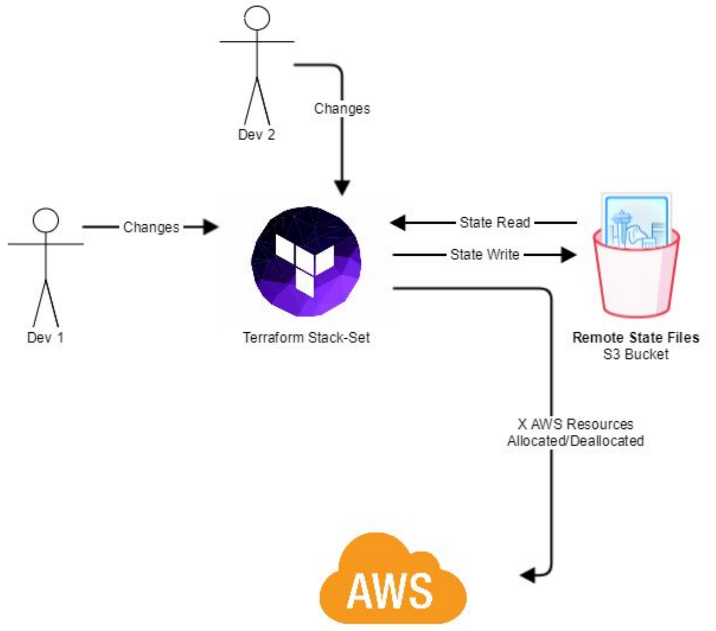
- Terraform depends on a state file
  - Maps real world resources to your configuration
  - Stored locally by default, easy to get out of sync



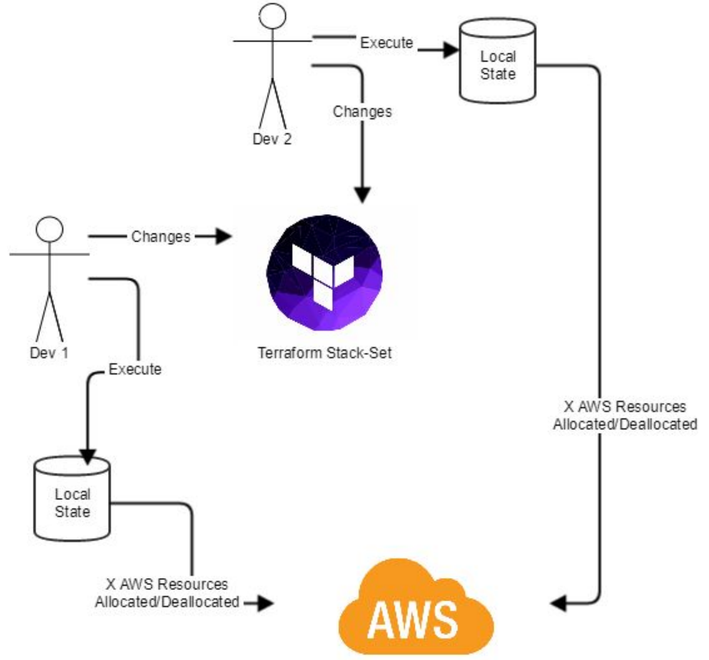
```
$ terraform state show aws_instance.demo_instance[0]
id                                          = i-00b1f4a9576fbf4a5
ami                                         = ami-8c1be5f6
associate_public_ip_address                 = true
availability_zone                           = us-east-1c
disable_api_termination                     = false
ebs_block_device.#                          = 0
ebs_optimized                               = false
ephemeral_block_device.#                    = 0
iam_instance_profile                        =
instance_state                              = running
instance_type                               = t2.micro
```

- Use remote state
  - Automatically syncs to a storage backend (Amazon S3) before and after terraform operations

Lessons Learned *State - Local vs Remote*



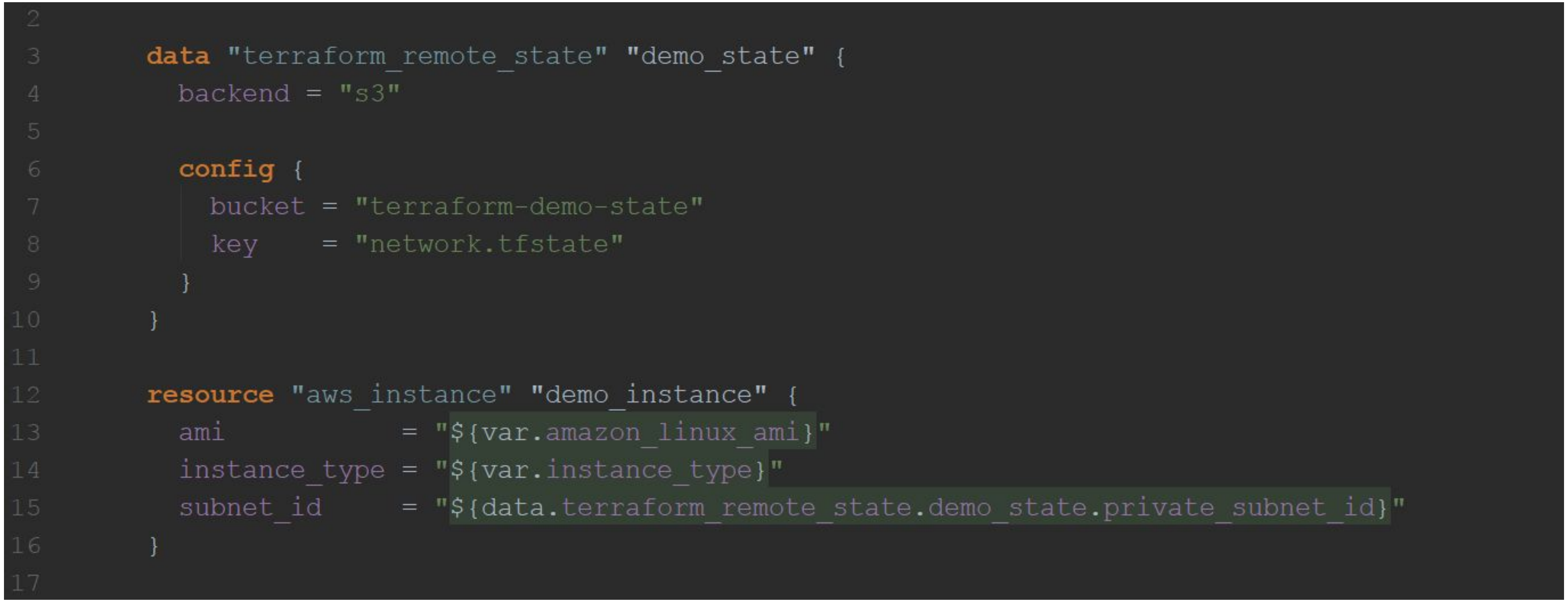
Rather Than



Lessons Learned *Organization*

- As codebase grows, avoid keeping everything in one state
    - Risk: each change can affect your entire environment
    - Consider breaking scripts into logical units that can be managed independently
        - Security roles, policies
        - Network
        - Application Server
    - Makes handling a mix of persistent/transient resources easier

Advanced Usage Example *Reading Outputs from a Remote State*



```
data "terraform_remote_state" "demo_state" {
  backend = "s3"

  config {
    bucket = "terraform-demo-state"
    key    = "network.tfstate"
  }
}

resource "aws_instance" "demo_instance" {
  ami           = "${var.amazon_linux_ami}"
  instance_type = "${var.instance_type}"
  subnet_id     = "${data.terraform_remote_state.demo_state.private_subnet_id}"
}
```

Any Questions?

- Getting Started: https://www.terraform.io/intro/getting-started/install.html
- Demo Source: https://github.com/csabatini/terraform-meetup-demo