E N T 3 5 6 - S

How Cox Automotive Runs GitHub Enterprise on AWS

Bryan Cross
Principal Solutions Engineer
GitHub

@bryancross

Felipe Ramirez
Senior Systems Engineer
Cox Automotive

@felipe-ramirez

AWS re:Invent

aws

Bryan Cross

Principal Solutions Engineer

GitHub

1,100,000,000

Contributions this year

31,000,000

Developers

100,000,000

Repositories

8,000,000

New developers

80%

Of users are from outside U.S.

What’s a Git?

What's a Git?

What's a Git?


git
/git/
noun BRITISH informal
an unpleasant or contemptible person.

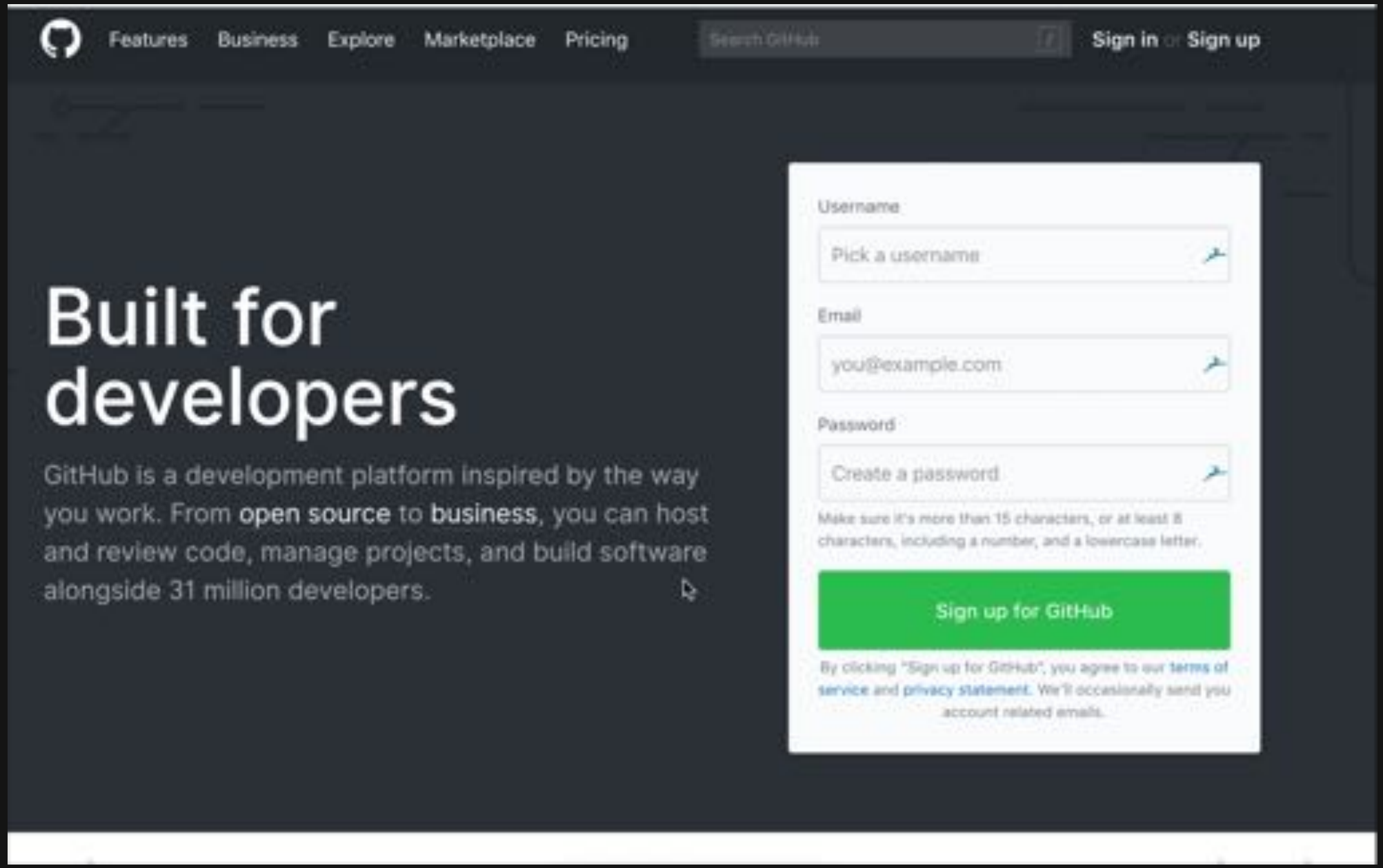
What's a Git?

"If you can't figure out who the Git on your team is, it's probably you"

- Your Team

$41,500,000,000

$41,500,000,000


Features
Business
Explore
Marketplace
Pricing
Search GitHub
Sign in or Sign up
Built for developers
GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.
Username
Pick a username
Email
you@example.com
Password
Create a password
Make sure it's more than 15 characters, or at least 8 characters, including a number, and a lowercase letter.
Sign up for GitHub
By clicking "Sign up for GitHub", you agree to our terms of service and privacy statement. We'll occasionally send you account related emails.

Same UX as GitHub

Scalable

Easy to manage/upgrade

Built in HA/DR

A true platform

Bryan Cross

Principal Solutions Engineer

GitHub

Bryan Cross

Principal Solutions Engineer

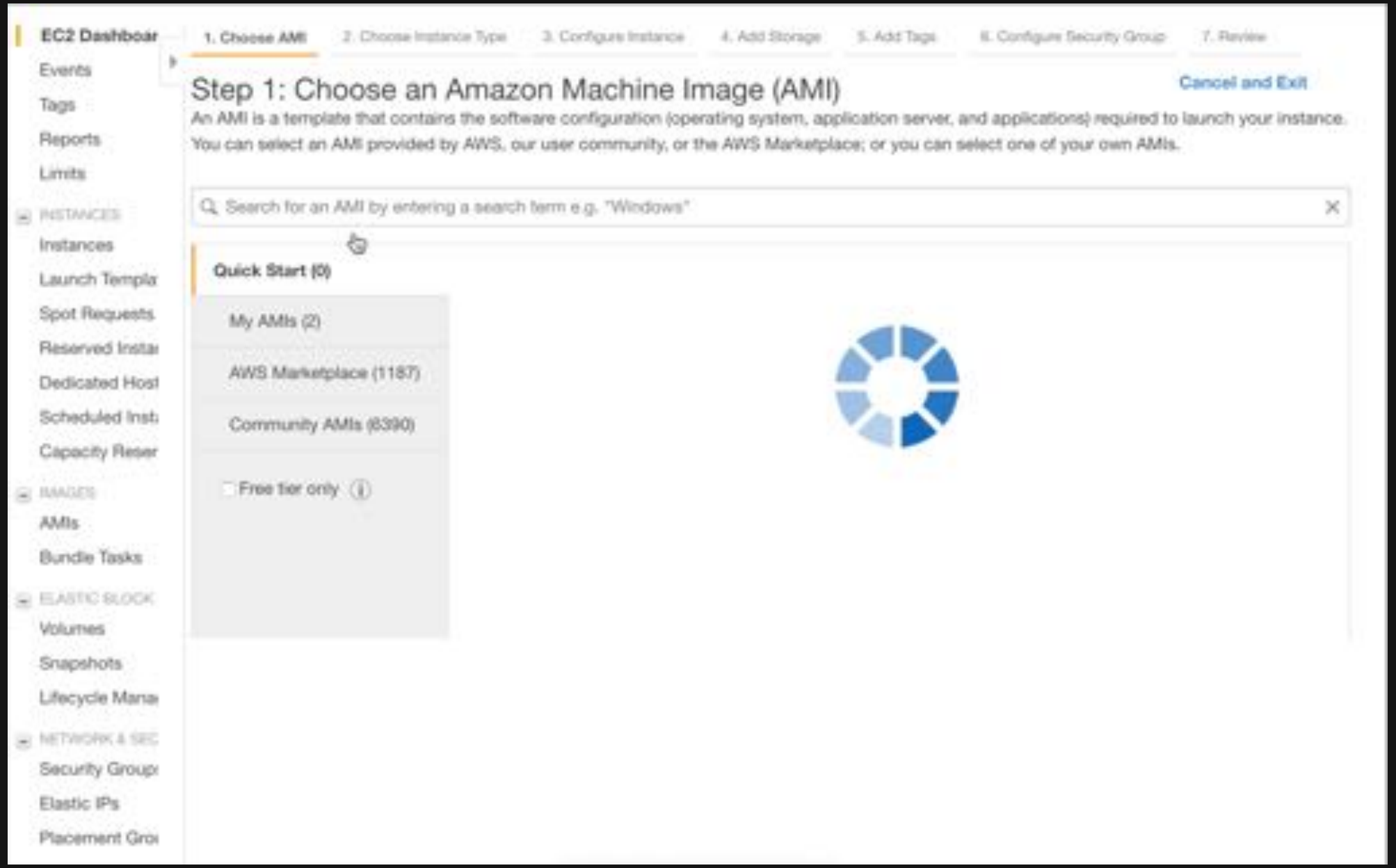
GitHub

Bryan Cross

Principal Solutions Engineer

GitHub

SALESGUY


EC2 Dashboar
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templa
Spot Requests
Reserved Insta
Dedicated Host
Scheduled Inst
Capacity Reser
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK
Volumes
Snapshots
Lifecycle Mana
NETWORK & SEC
Security Group
Elastic IPs
Placement Gro
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance.
You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
Search for an AMI by entering a search term e.g. "Windows"
Quick Start (0)
My AMIs (2)
AWS Marketplace (1187)
Community AMIs (8390)
Free tier only


THIS BAD BOY CAN HOLD
GitHub Enterprise
SO MANY REPOS

Felipe Ramirez

Senior Systems Engineer

Cox Automotive

COX AUTOMOTIVE™

INVENTORY SOLUTIONS

Manheim

RMS/AUTOMOTIVE

RETAIL SOLUTIONS

HOMENET AUTOMOTIVE

VinSolutions

DEALER.COM

xtime

Dealertrack dt

vAuto LIVE MARKET VIEW

Kelley Blue Book

Autotrader

FINANCIAL SOLUTIONS

NEXTGEAR CAPITAL

MOBILITY SOLUTIONS

Getaround

RIDECELL

CLUTCH

FLEXDRIVE

OUSTER

INTERNATIONAL

AUTOST. 汽车街 autostreets.com

易车 yiche.com

carsguide

Dealer-Auction.com

DEALER SOLUTIONS

精真估 jingzhengu.com

Mahindra First Choice

Modix

MOLICAR

MOTORS .co.uk

MOVEX

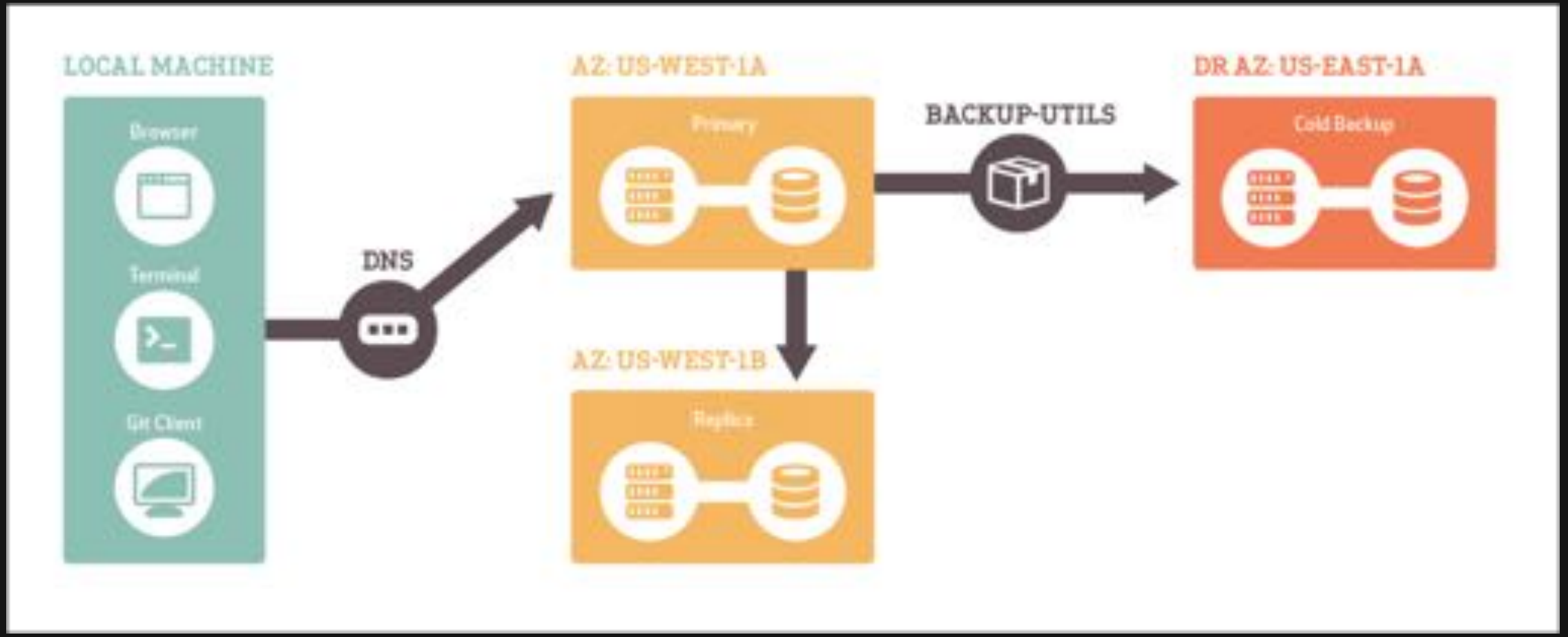
github.com/felipe-ramirez/reinvent18

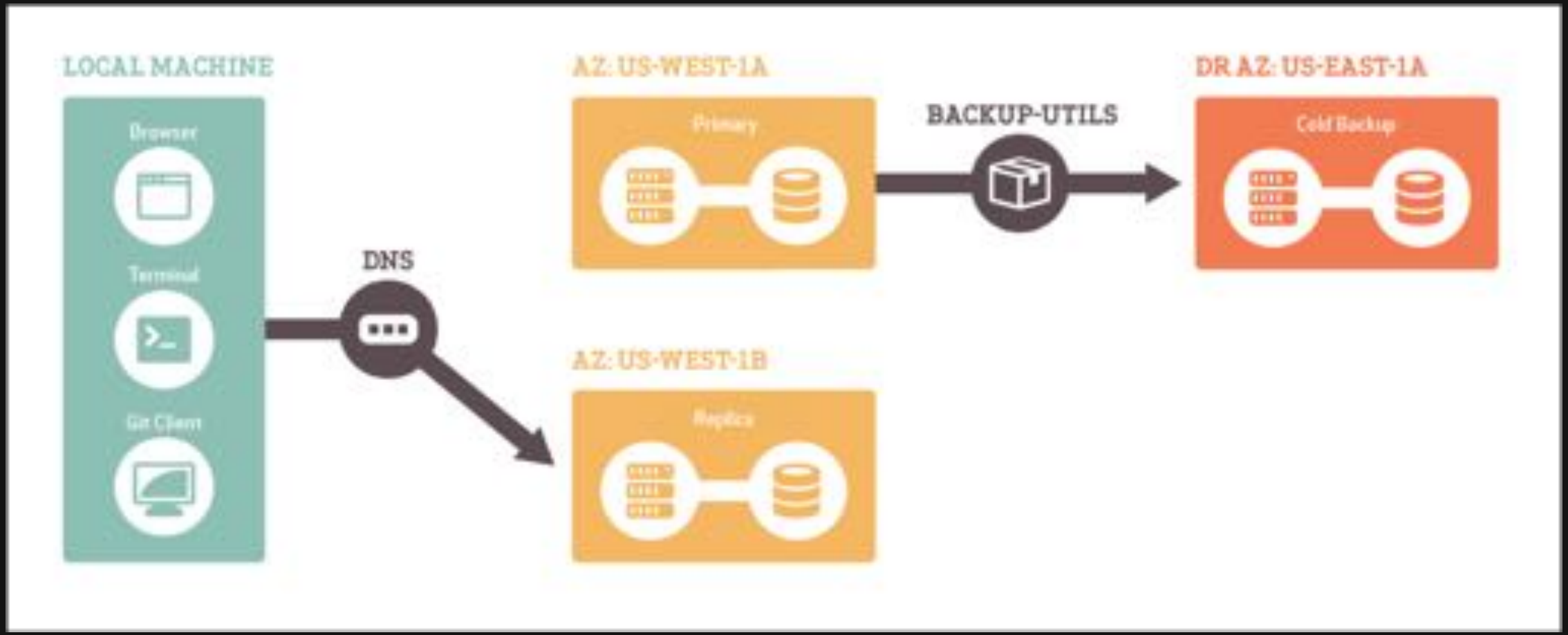
Agenda

1. GitHub Enterprise HA on AWS
2. Defining infrastructure with Terraform
3. Using API & Terraform for configuration
4. Future enhancements

Agenda

1. GitHub Enterprise HA on AWS
2. Defining infrastructure with Terraform
3. Using API & Terraform for configuration
4. Future enhancements


LOCAL MACHINE
Browser
Terminal
Git Client
DNS
AZ: US-WEST-1A
Primary
AZ: US-WEST-1B
Replica
BACKUP-UTILS
DR AZ: US-EAST-1A
Cold Backup


LOCAL MACHINE
Browser
Terminal
Git Client
DNS
AZ: US-WEST-1A
Primary
AZ: US-WEST-1B
Replica
BACKUP-UTILS
DR AZ: US-EAST-1A
Cold Backup

HA doesn’t replace backups

github.com/github/backup-utils

Agenda

1. GitHub Enterprise HA on AWS
2. Defining infrastructure with Terraform
3. Using API & Terraform for configuration
4. Future enhancements

Why deploy GitHub Enterprise with code?

Code as documentation

Consistency and speed

It’s fun to do

HashiCorp

Terraform

GitHub Enterprise AMI ID (ami.tf)

```
data "aws_ami" "ghe" {
  filter {
    name   = "name"
    values = ["GitHub Enterprise ${var.ghe_version}"]
  }

  owners = ["895557238572"]
}
```

GitHub Enterprise AMI ID (ami.tf)

```
data "aws_ami" "ghe" {
  filter {
    name   = "name"
    values = ["GitHub Enterprise ${var.ghe_version}"]
  }

  owners = ["895557238572"]
}
```

GitHub Enterprise AMI ID (ami.tf)

```
data "aws_ami" "ghe" {
  filter {
    name   = "name"
    values = ["GitHub Enterprise ${var.ghe_version}"]
  }

  owners = ["895557238572"]
}
```

EC2 instances (main.tf)

```
resource "aws_instance" "ghe" {
  count                  = 2
  ami                    = "${data.aws_ami.ghe.id}"
  subnet_id              = "${element(var.subnet_ids, count.index)}"
  instance_type          = "${var.instance_type}"
  vpc_security_group_ids = ["${aws_security_group.ghe.id}"]

  ebs_block_device {
    device_name = "/dev/xvdg"
    volume_type = "gp2"
    volume_size = "${var.data_volume_size}"
  }

  tags = {
    Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
  }
}
```

EC2 instances (main.tf)

```
resource "aws_instance" "ghe" {
  count                  = 2
  ami                    = "${data.aws_ami.ghe.id}"
  subnet_id              = "${element(var.subnet_ids, count.index)}"
  instance_type          = "${var.instance_type}"
  vpc_security_group_ids = ["${aws_security_group.ghe.id}"]

  ebs_block_device {
    device_name = "/dev/xvdg"
    volume_type = "gp2"
    volume_size = "${var.data_volume_size}"
  }

  tags = {
    Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
  }
}
```

EC2 instances (main.tf)

```
resource "aws_instance" "ghe" {
  count                  = 2
  ami                    = "${data.aws_ami.ghe.id}"
  subnet_id              = "${element(var.subnet_ids, count.index)}"
  instance_type          = "${var.instance_type}"
  vpc_security_group_ids = ["${aws_security_group.ghe.id}"]

  ebs_block_device {
    device_name = "/dev/xvdg"
    volume_type = "gp2"
    volume_size = "${var.data_volume_size}"
  }

  tags = {
    Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
  }
}
```

EC2 instances (main.tf)

```
resource "aws_instance" "ghe" {
  count                  = 2
  ami                    = "${data.aws_ami.ghe.id}"
  subnet_id              = "${element(var.subnet_ids, count.index)}"
  instance_type          = "${var.instance_type}"
  vpc_security_group_ids = ["${aws_security_group.ghe.id}"]

  ebs_block_device {
    device_name = "/dev/xvdg"
    volume_type = "gp2"
    volume_size = "${var.data_volume_size}"
  }

  tags = {
    Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
  }
}
```

EC2 instances (main.tf)

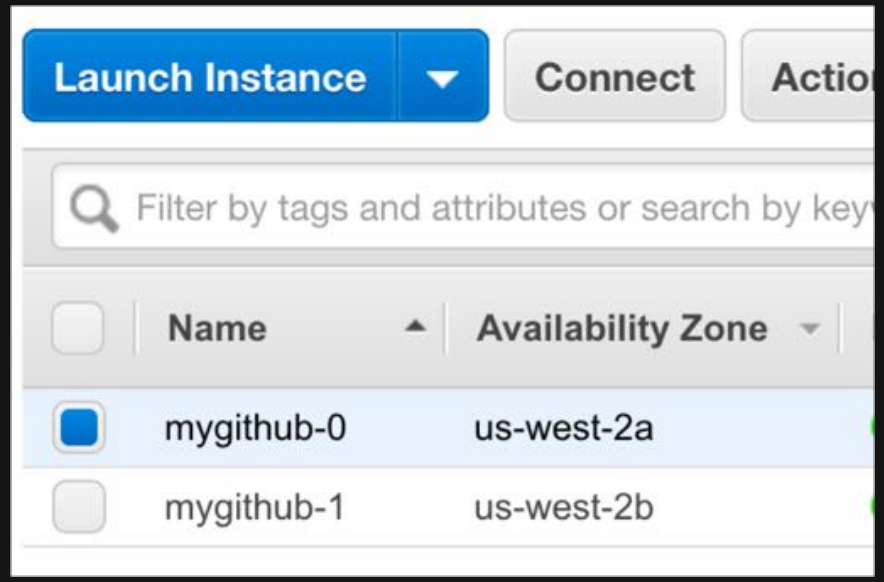
```
resource "aws_instance" "ghe" {
  count                  = 2
  ami                    = "${data.aws_ami.ghe.id}"
  subnet_id              = "${element(var.subnet_ids, count.index)}"
  instance_type          = "${var.instance_type}"
  vpc_security_group_ids = ["${aws_security_group.ghe.id}"]

  ebs_block_device {
    device_name = "/dev/xvdg"
    volume_type = "gp2"
    volume_size = "${var.data_volume_size}"
  }

  tags = {
    Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
  }
}
```

EC2 instances Name tag

mygithub.example.com



EC2 instances Name tag (main.tf)

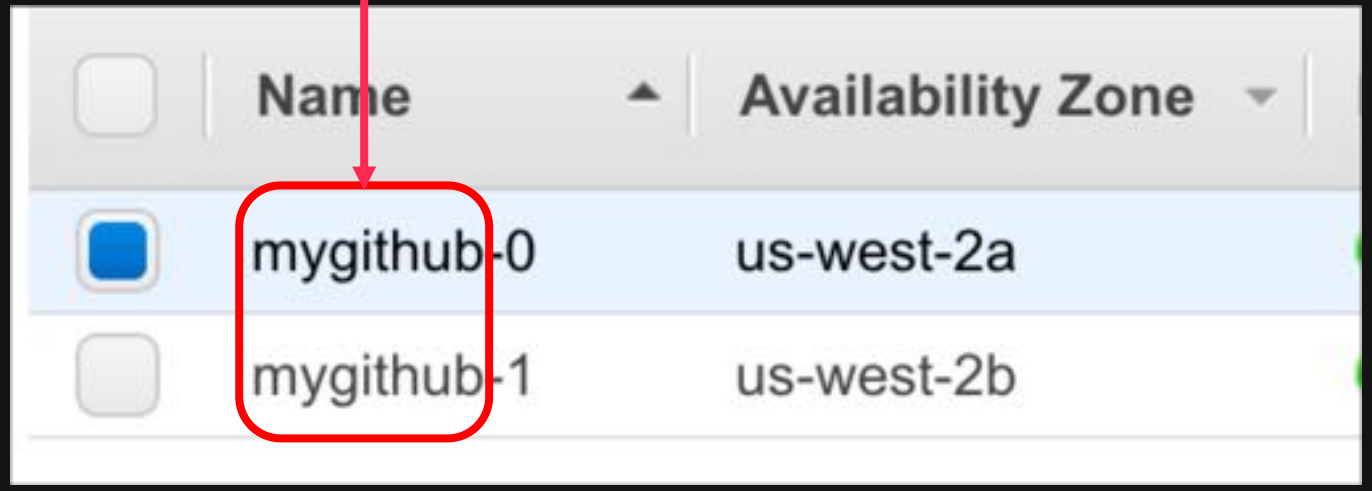
```
locals {
  env_tag = "${element(split(".", var.ghe_hostname), 0)}"
}
```

EC2 instances Name tag (main.tf)

```
locals {
  env_tag = "${element(split(".", var.ghe_hostname), 0)}"
}
```

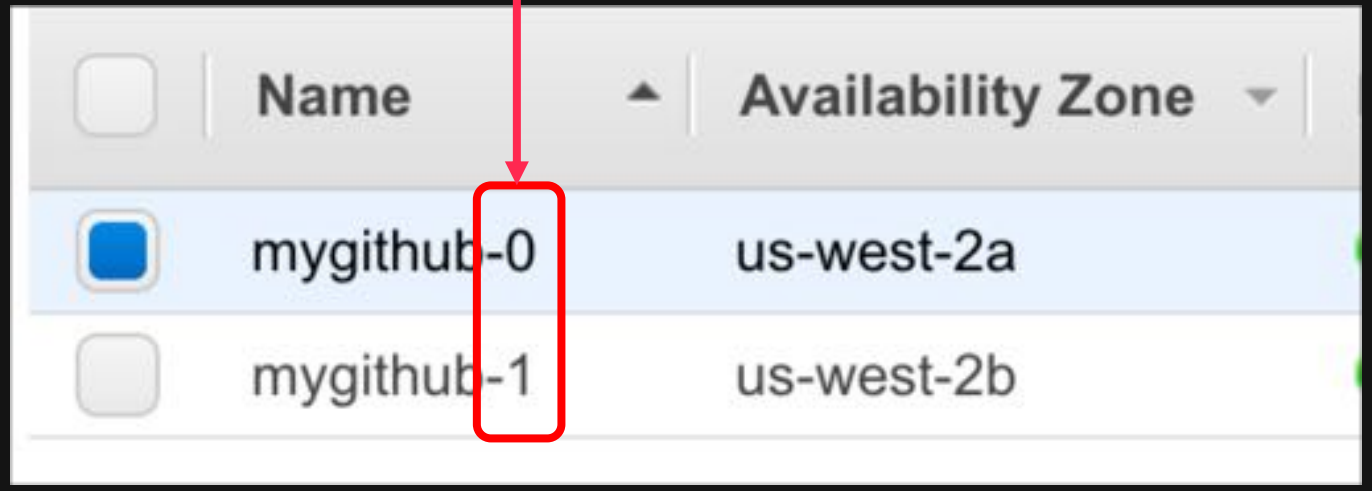
EC2 instances Name tag (main.tf)

```
tags = {
  Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
}
```



EC2 instances Name tag (main.tf)

```
tags = {
  Name = "${local.env_tag}-${format("%01d", count.index + 0)}"
}
```



Security Group and Rules (sg.tf)

```
resource "aws_security_group" "ghe" {
  name        = "ghe-sg"
  vpc_id      = "${var.vpc_id}"
  description = "GitHub Enterprise Appliance Security Group"

  tags = {
    Name      = "${local.env_tag}-appliance SG"
  }
}

resource "aws_security_group_rule" "allow_ssh" {
  type              = "ingress"
  from_port         = 22
  to_port           = 22
  protocol          = "tcp"
  cidr_blocks       = ["0.0.0.0/0"]
  security_group_id = "${aws_security_group.ghe.id}"
}

...
```

Elastic IP addresses (eip.tf)

```
resource "aws_eip" "ghe_appliance" {
  count    = 2
  instance = "${element(aws_instance.ghe_appliance.*.id, count.index)}"
  vpc      = true
}
```

Route 53 DNS (r53.tf)

```
resource "aws_route53_zone" "ghe" {
  name = "${var.ghe_hostname}"
}

resource "aws_route53_record" "ghe_hostname" {
  zone_id = "${aws_route53_zone.ghe.zone_id}"
  name    = "${var.ghe_hostname}"
  type    = "A"
  ttl     = "300"
  records = ["${var.primary == "0" ? aws_eip.ghe.0.public_ip : aws_eip.ghe.1.public_ip}"]
}

resource "aws_route53_record" "ghe_wildcard" {
  zone_id = "${aws_route53_zone.ghe.zone_id}"
  name    = "*.${var.ghe_hostname}"
  type    = "A"
  ttl     = "300"
  records = ["${var.primary == "0" ? aws_eip.ghe.0.public_ip : aws_eip.ghe.1.public_ip}"]
}
```

Route 53 DNS (r53.tf)

```
records = ["${var.primary == "0" ? aws_eip.ghe.0.public_ip : aws_eip.ghe.1.public_ip}"]
```

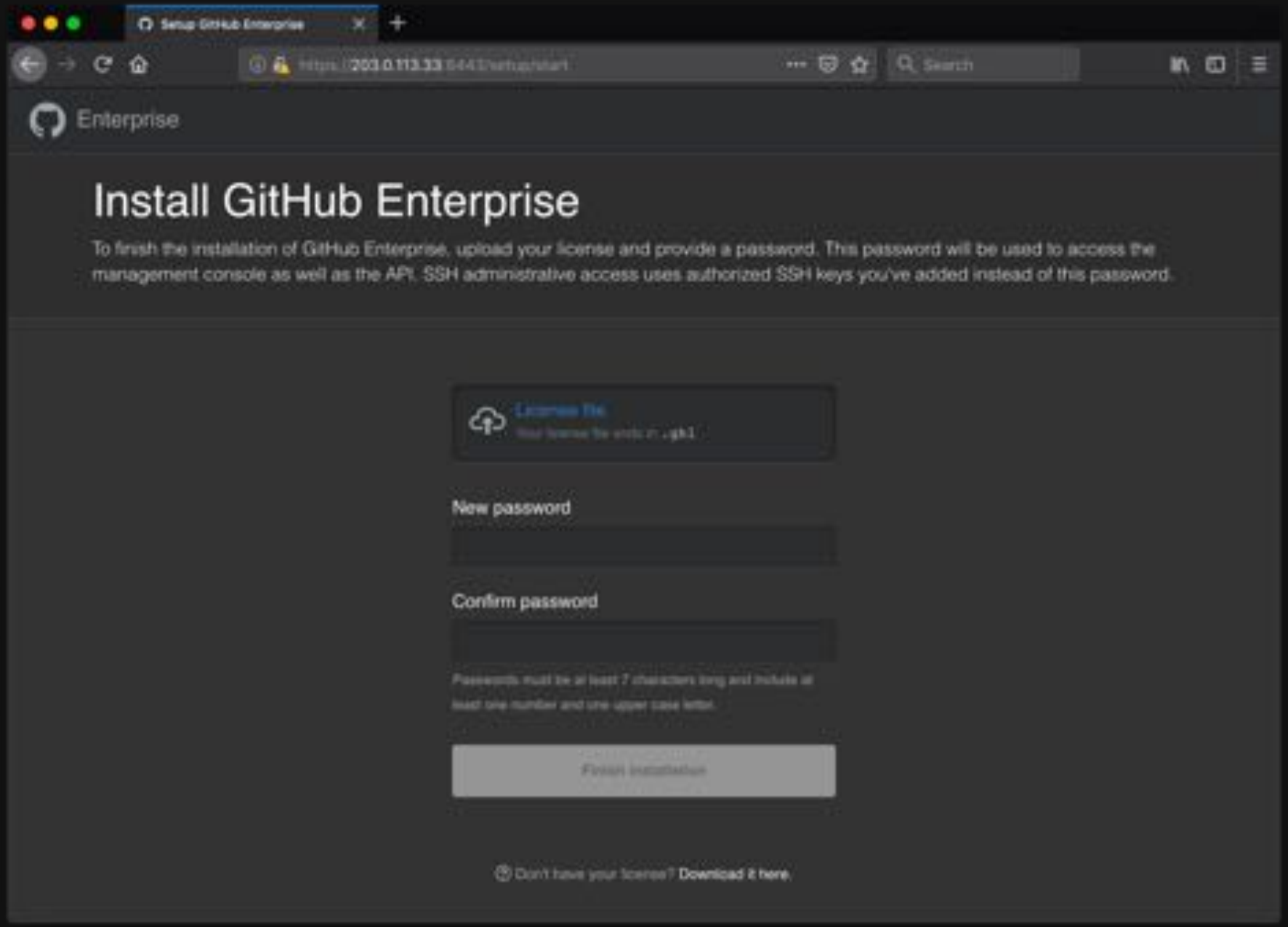
condition

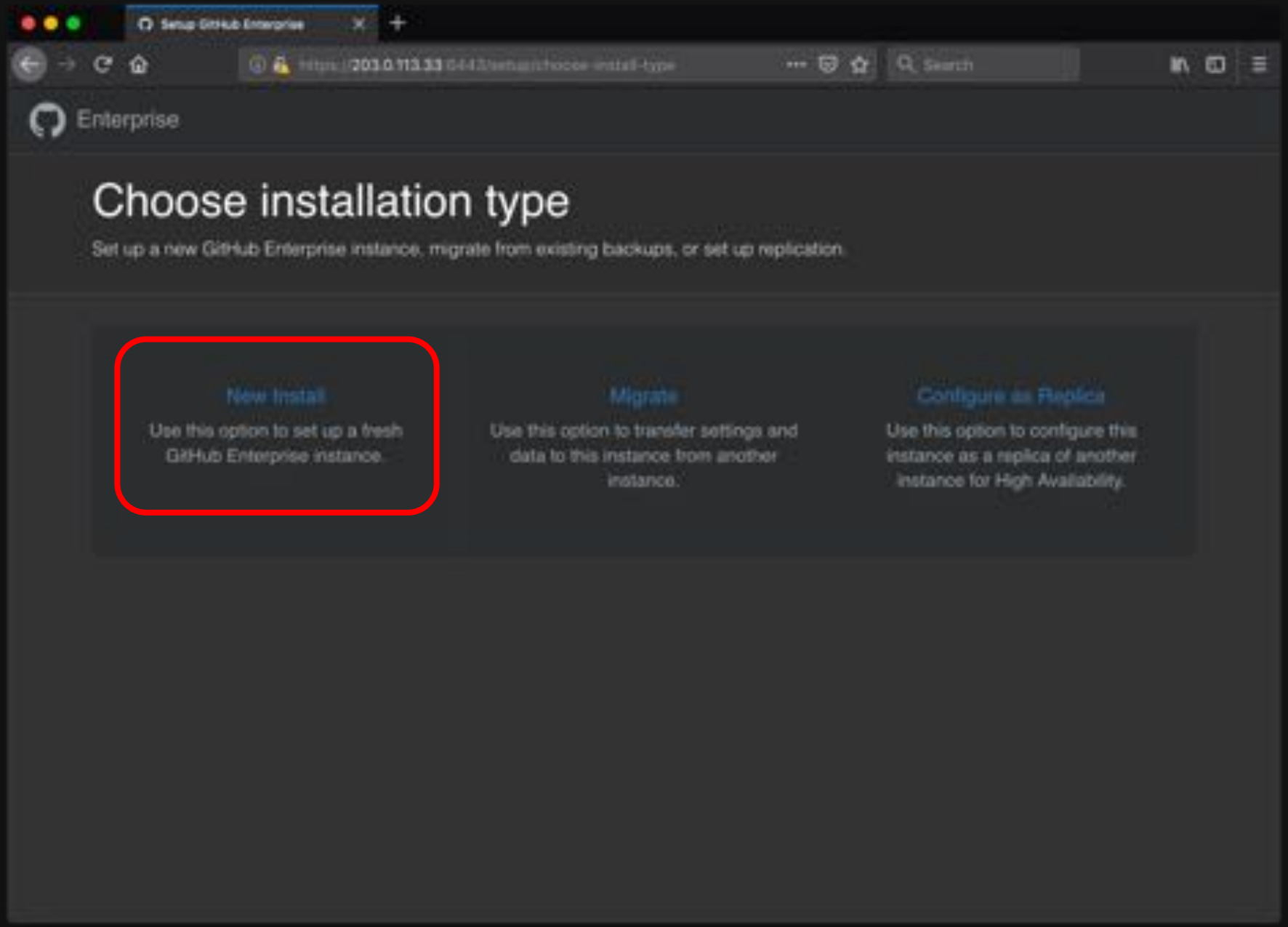
true value

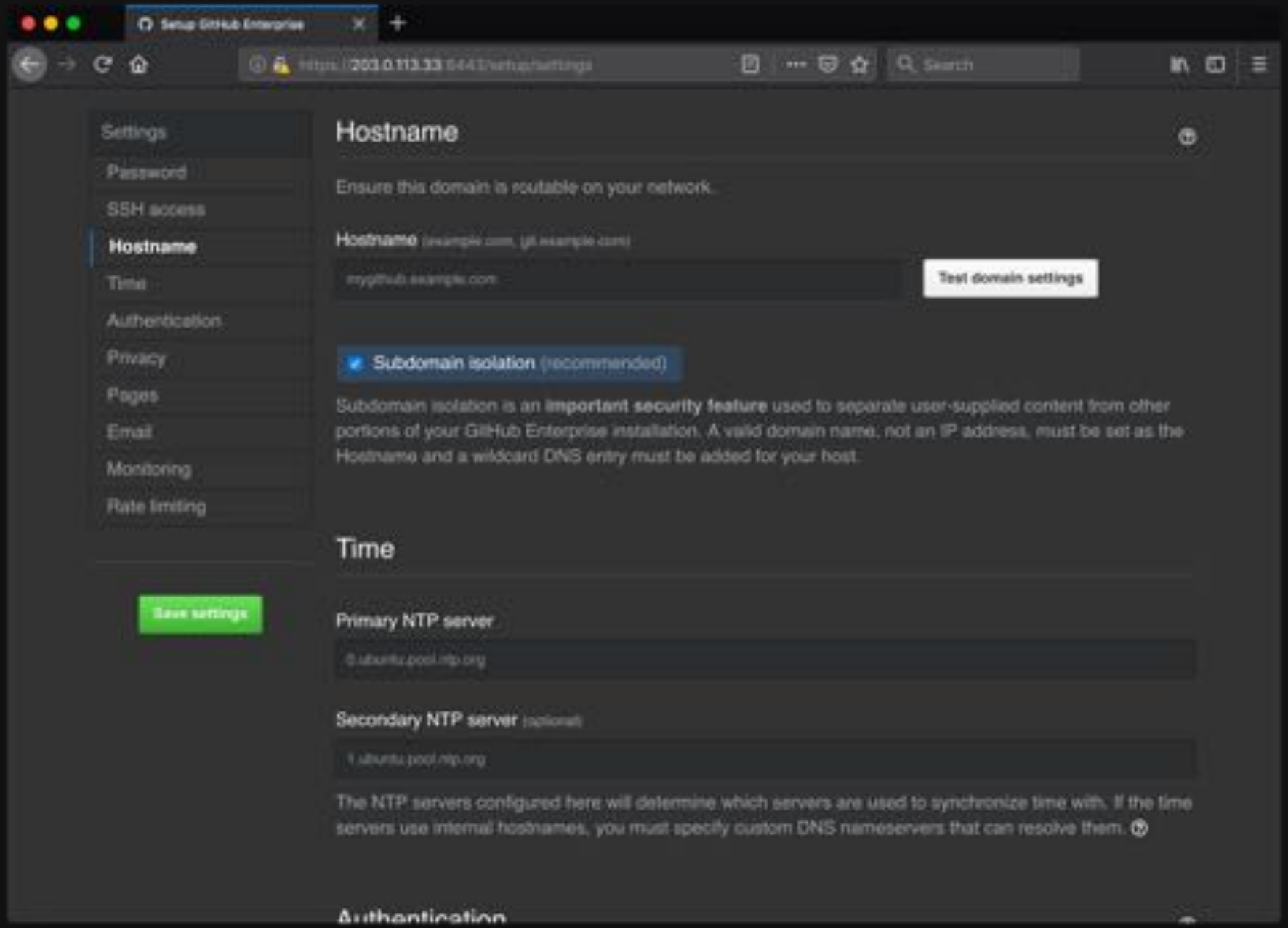
false value

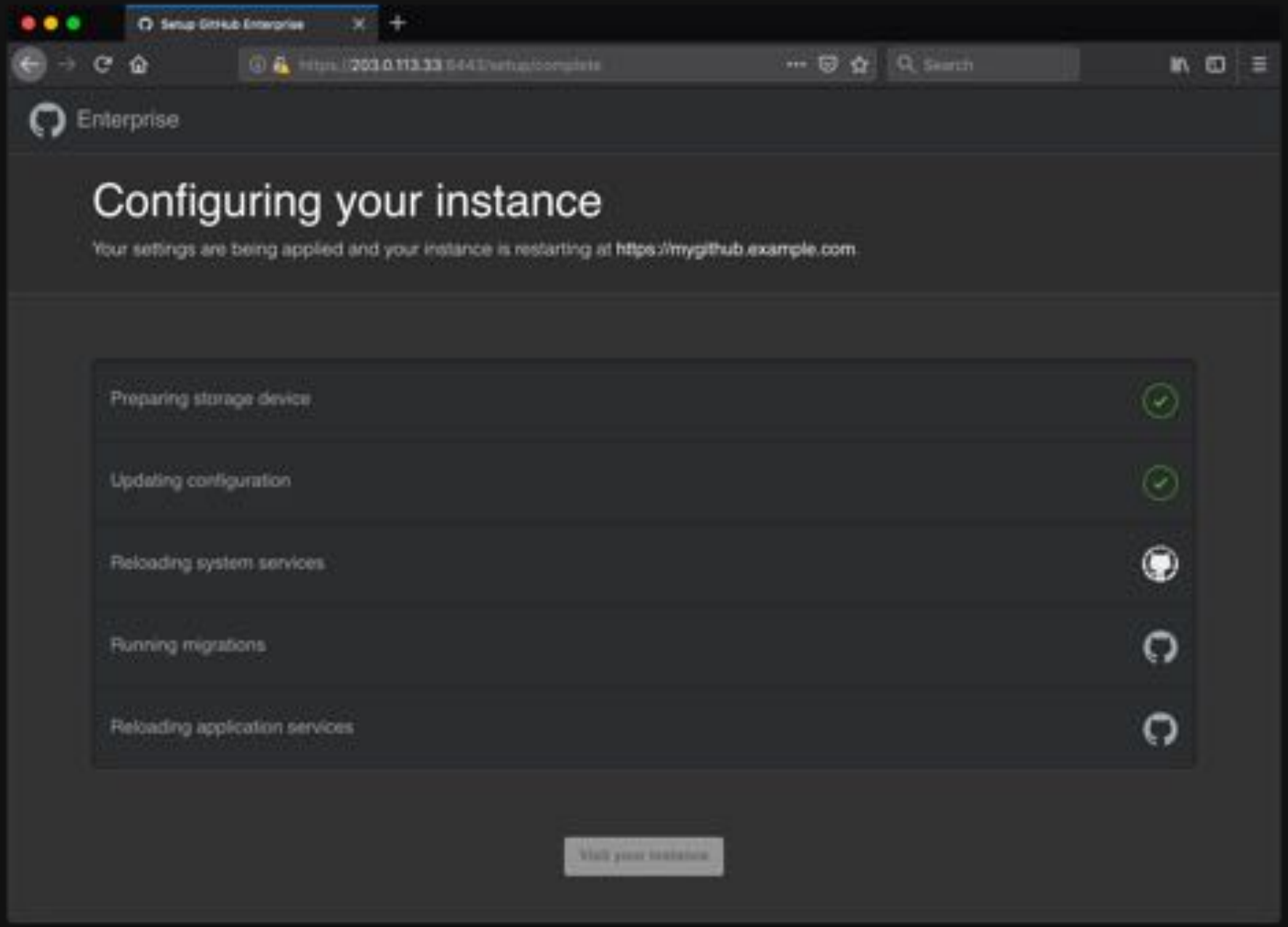
Agenda

1. GitHub Enterprise HA on AWS
2. Defining infrastructure with Terraform
3. Using API & Terraform for configuration
4. Future enhancements


Setup GitHub Enterprise
Search
Enterprise
Install GitHub Enterprise
To finish the installation of GitHub Enterprise, upload your license and provide a password. This password will be used to access the management console as well as the API. SSH administrative access uses authorized SSH keys you've added instead of this password.
License file
.ghl
New password
Confirm password
Finish installation
Don't have your license? Download it here.


Setup GitHub Enterprise
Enterprise
Choose installation type
Set up a new GitHub Enterprise instance, migrate from existing backups, or set up replication.
New Install
Use this option to set up a fresh GitHub Enterprise instance.
Migrate
Use this option to transfer settings and data to this instance from another instance.
Configure as Replica
Use this option to configure this instance as a replica of another instance for High Availability.


Setup GitHub Enterprise
https://203.0.113.33:8443/setup/settings
Search
Settings
Password
SSH access
Hostname
Time
Authentication
Privacy
Pages
Email
Monitoring
Rate limiting
Save settings
Hostname
Ensure this domain is routable on your network.
Hostname (example.com, git.example.com)
mygithub.example.com
Test domain settings
Subdomain isolation (recommended)
Subdomain isolation is an important security feature used to separate user-supplied content from other portions of your GitHub Enterprise installation. A valid domain name, not an IP address, must be set as the Hostname and a wildcard DNS entry must be added for your host.
Time
Primary NTP server
0.ubuntu.pool.ntp.org
Secondary NTP server (optional)
1.ubuntu.pool.ntp.org
The NTP servers configured here will determine which servers are used to synchronize time with. If the time servers use internal hostnames, you must specify custom DNS nameservers that can resolve them.
Authentication


Setup GitHub Enterprise
https://203.0.113.33:8443/setup/complete
Search
Enterprise
Configuring your instance
Your settings are being applied and your instance is restarting at https://mygithub.example.com
Preparing storage device
Updating configuration
Reloading system services
Running migrations
Reloading application services
Visit your instance

Existing appliance settings export (API)

```
curl -sL 'https://api_key:Octocat1@github.example.com:8443/setup/api/settings' \
  >settings.json
```

Existing appliance settings export (API)


```
{
  "enterprise": {
    "private_mode": true,
    "public_pages": false,
    "subdomain_isolation": true,
    "signup_enabled": false,
    "github_hostname": "github.example.com",
    "identicons_host": "dotcom",
    "http_proxy": null,
    "http_noproxy": null,
    "auth_mode": "saml",
    "builtin_auth_fallback": false,
    "expire_sessions": false,
    "admin_password": null,
    "configuration_id": 1234565432,
    "configuration_run_count": 5,
    "avatar": null,
    ...
```

Verify the new appliance is ready

```
curl -skL 'https://203.0.113.33:8443/setup/start' \
  | grep "Setup GitHub Enterprise"
```

Verify the new appliance is ready

```
curl -skL 'https://203.0.113.33:8443/setup/start' \
  | grep "Setup GitHub Enterprise"
```

Upload license and set a password (API)

```
curl -skL -X POST 'https://203.0.113.33:8443/setup/api/start' \
  -F license=@/tmp/github-enterprise.ghl \
  -F "password=reInvent18"
```

Modify new appliance settings (API)

```
curl -skL -X PUT 'https://api_key:reInvent18@203.0.113.33:8443/setup/api/settings' \
  --data-urlencode "settings=`cat /tmp/settings.json`"
```

Start a configuration process (API)

```
curl -skL -X POST 'https://api_key:reInvent18@203.0.113.33:8443/setup/api/configure'
```

AWS Systems Manager > Parameter store >
Create parameter

Parameter details

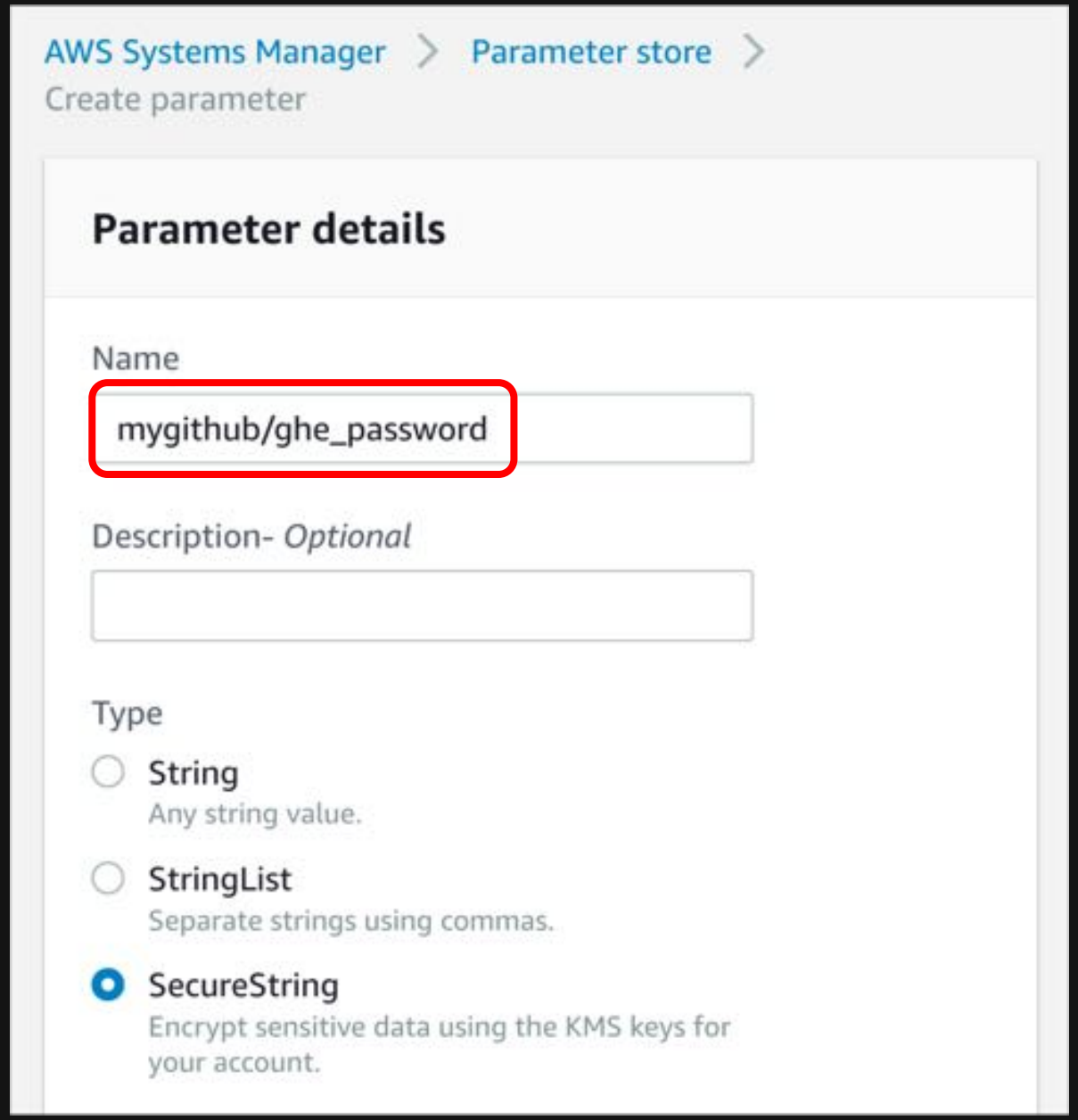
Name

mygithub/ghe_password

Description- *Optional*

Type

- String
  Any string value.
- StringList
  Separate strings using commas.
- SecureString
  Encrypt sensitive data using the KMS keys for your account.


AWS Systems Manager
Parameter store
Create parameter
Parameter details
Name
mygithub/ghe_password
Description- Optional
Type
String
Any string value.
StringList
Separate strings using commas.
SecureString
Encrypt sensitive data using the KMS keys for your account.

Get Parameter Store password (main.tf)

```
data "aws_ssm_parameter" "ghe_password" {
  name = "/${local.env_tag}/ghe_password"
}
```

Get Parameter Store password (main.tf)

```
locals {
  env_tag      = "${element(split(".", var.ghe_hostname), 0)}"
→ ghe_password = "${data.aws_ssm_parameter.ghe_password.value}"
}
```

Execute API commands (main.tf)

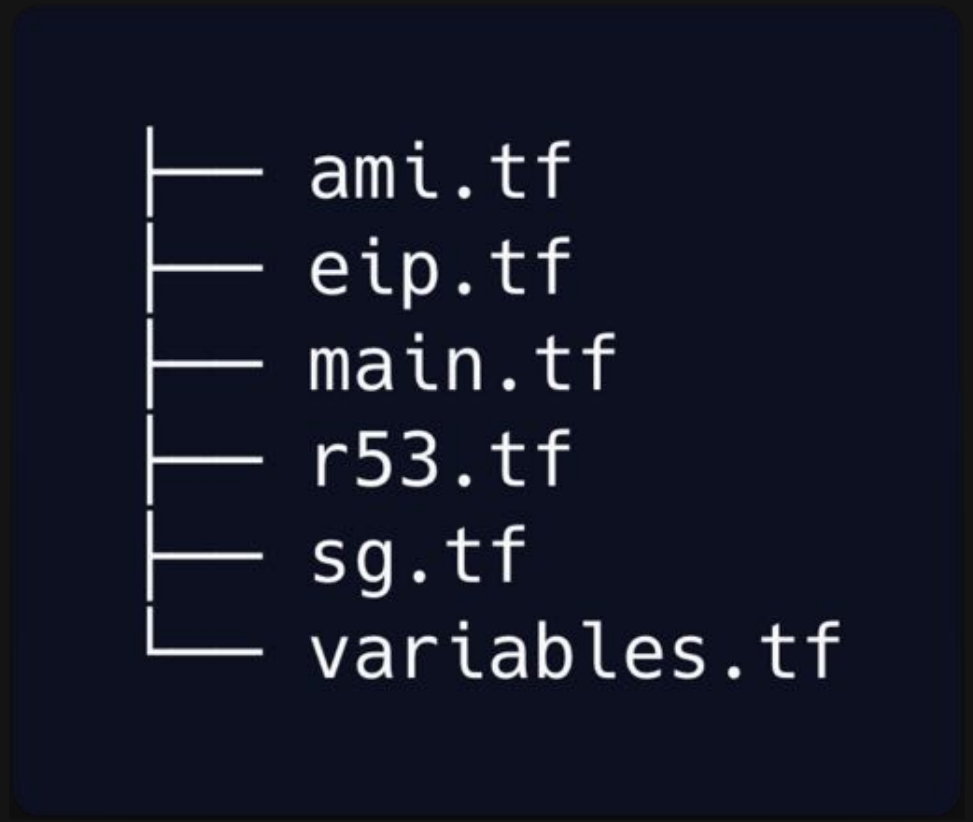
```
resource "null_resource" "ghe_config" {
  // Check that GHE is ready to take API requests
  provisioner "local-exec" {
    command = "until | $(curl -skL 'https://${aws_eip.ghe.0.public_ip}:8443/setup/start' | grep -o 'Setup GitHub...
  }

  // Upload license and set Management Console password
  provisioner "local-exec" {
    command = "sleep 5 && curl -skL -X POST 'https://${aws_eip.ghe.0.public_ip}:8443/setup/api/start' -F license...
  }

  // Upload JSON configuration
  provisioner "local-exec" {
    command = "sleep 10 && curl -skL -X PUT 'https://api_key:${local.ghe_password}@${aws_eip.ghe.0.public_ip}:84...
  }

  // Start configuring
  provisioner "local-exec" {
    command = "sleep 10 && curl -skL -X POST 'https://api_key:${local.ghe_password}@${aws_eip.ghe.0.public_ip}:8...
  }
}
```

Module repo contents



”live” repo contents

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

```
provider "aws" {
  region = "us-west-2"
}

module "ghe" {
  source = "git::ssh://git@github.com/felipe-ramirez/reinvent18?ref=v0.1.0"

  ghe_version  = "2.15.1"
  ghe_hostname = "mygithub.example.com"
  ghe_license  = "github-enterprise.ghl"
  ghe_settings = "settings.json"

  primary = "0"

  vpc_id       = "vpc-123b1234"
  subnet_ids   = ["subnet-546c8a45", "subnet-23c45d6e"]

  instance_type    = "r4.large"
  data_volume_size = "500"
}
```

Agenda

1. GitHub Enterprise HA on AWS
2. Defining infrastructure with Terraform
3. Using API & Terraform for configuration
4. Future enhancements

Automate HA configuration on replica appliance

Use AWS CodePipeline for applying updates

Automated daily testing of backup data

github.com/felipe-ramirez/reinvent18

Thank you!

Bryan Cross
bryancross@github.com

Felipe Ramirez
felipe.ramirez@coxautoinc.com

AWS re:Invent

aws