

Understanding What DevOps Replaces



Chris B. Behrens
Senior Software Developer

@chrisbbehrrens



Architecture and Technology Fundamentals of DevOps



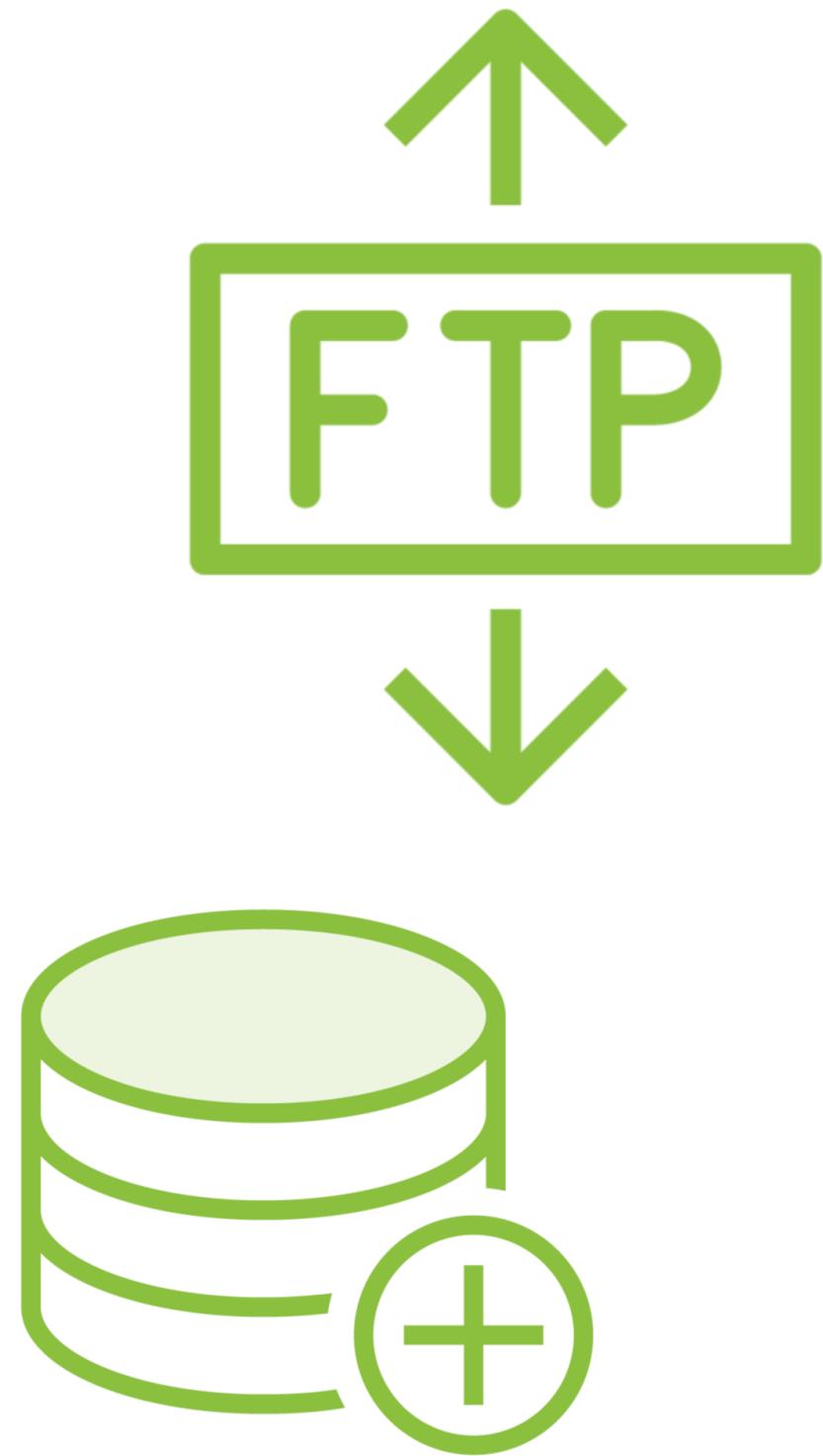
The Sample Architecture

Everything here applies broadly, but...

Let's create some specifics for storytelling



A Release Cycle



Good Enough

**For one-man operations, this
is fine**

Works 70% of the time



The Effect of the Second Developer

Parallel development

There is a merge conflict

Deployment is manual and tricky

Developer B's changes are missing

Because the merge was late, the merge is poorly tested

And we haven't accounted for the schema change



The Atomic Unit of DevOps – the Build



What Code?



Let's assume that we've at least got the code in version control

Features are developed in feature branches off of the master

Deployments happen from the dev's machine

So, it's whatever branch is checked out at that moment



The code is built on the branch
which is designated for
deployment.



What This Accomplishes

The decision alone cures several problems

This forces a better version control model

The build is a poka-yoke

Now, there's no way to deploy the wrong branch



The Effect on Version Control



All this feeds back into version control practices

“You merge too often”

All the merge conflicts landed on his plate

He thought this was discourteous

“Why don’t you do this?”

Things would be better if you did

This is Deliver Fast



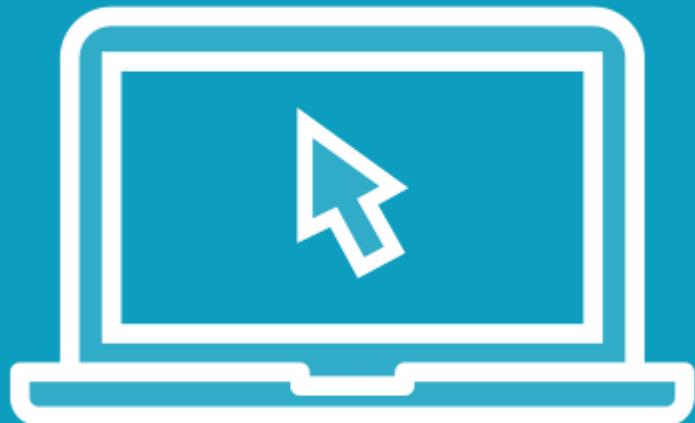
So, How Do I Create a Build?

A Build engine

A Build script



Demo



Create the simplest possible code

Create the simplest possible build script

Take a quick look at a real build engine

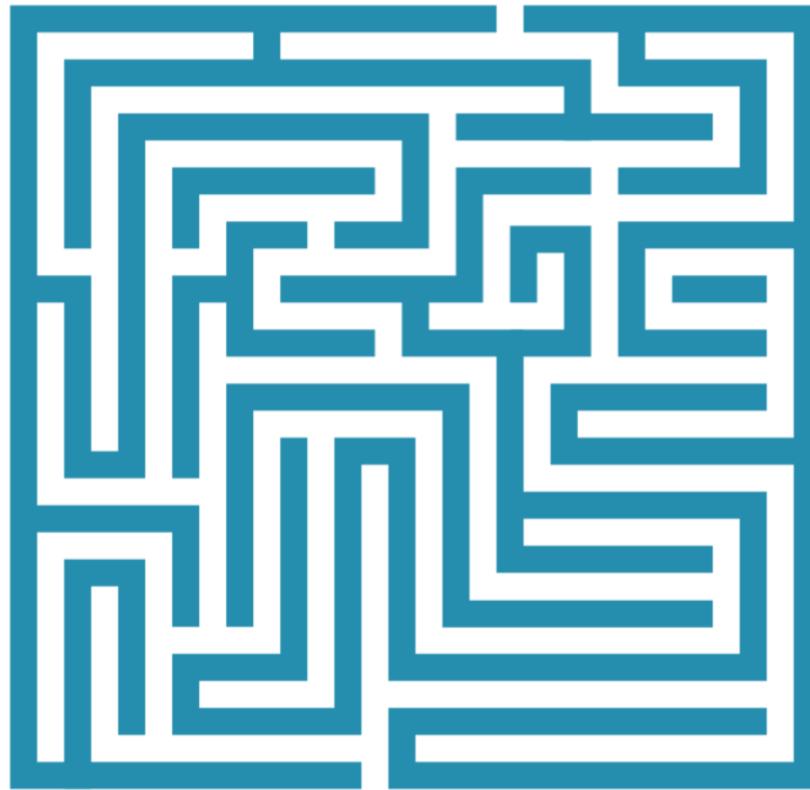
Azure DevOps Pipelines



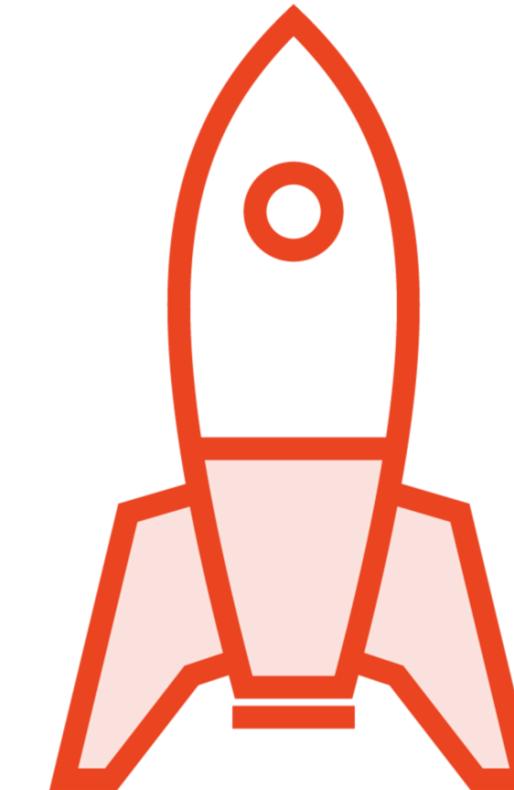
The Architecture That Facilitates DevOps



Let's Get a Little Ahead of Ourselves



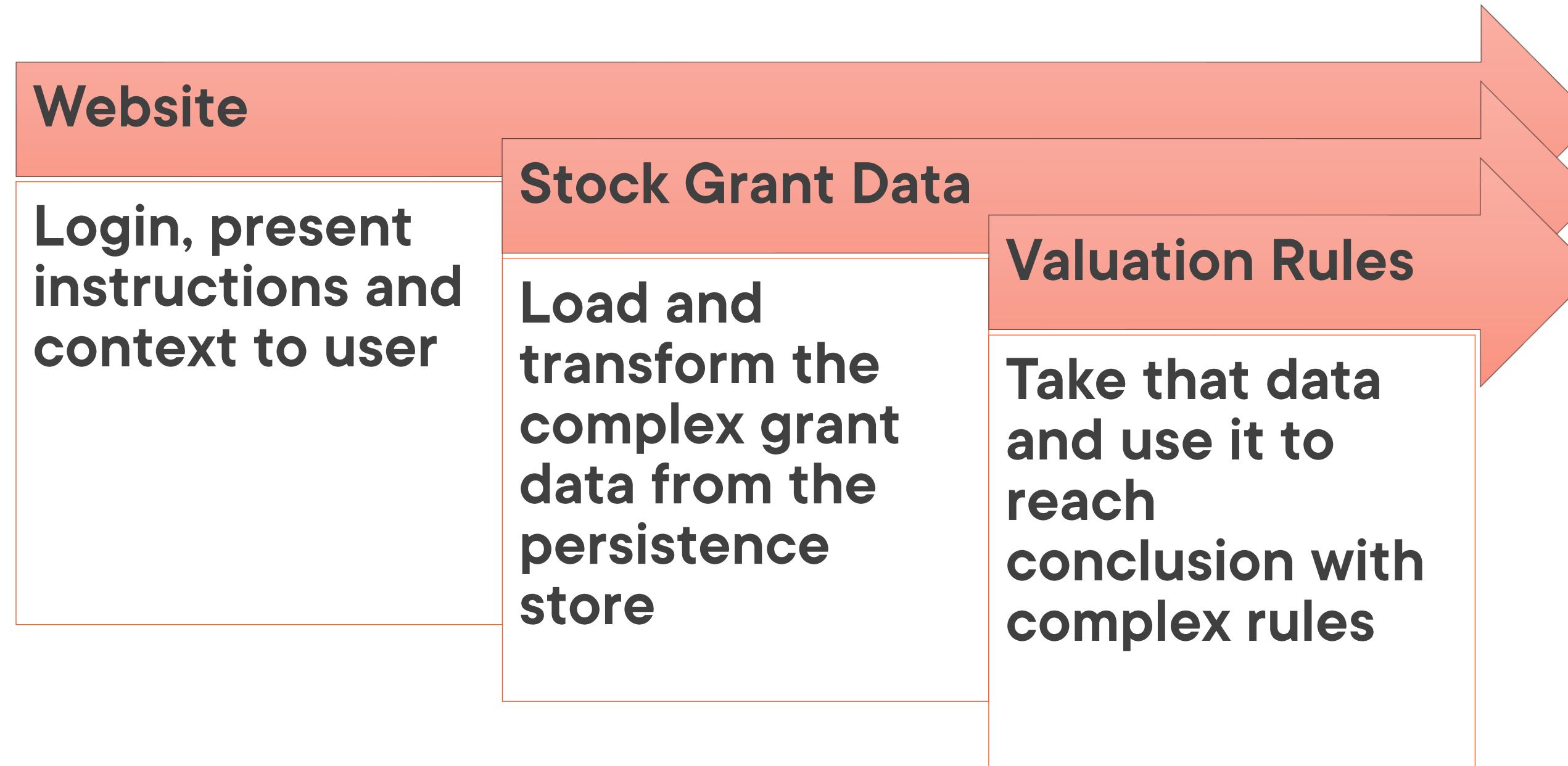
**What if our application was very
complex?**



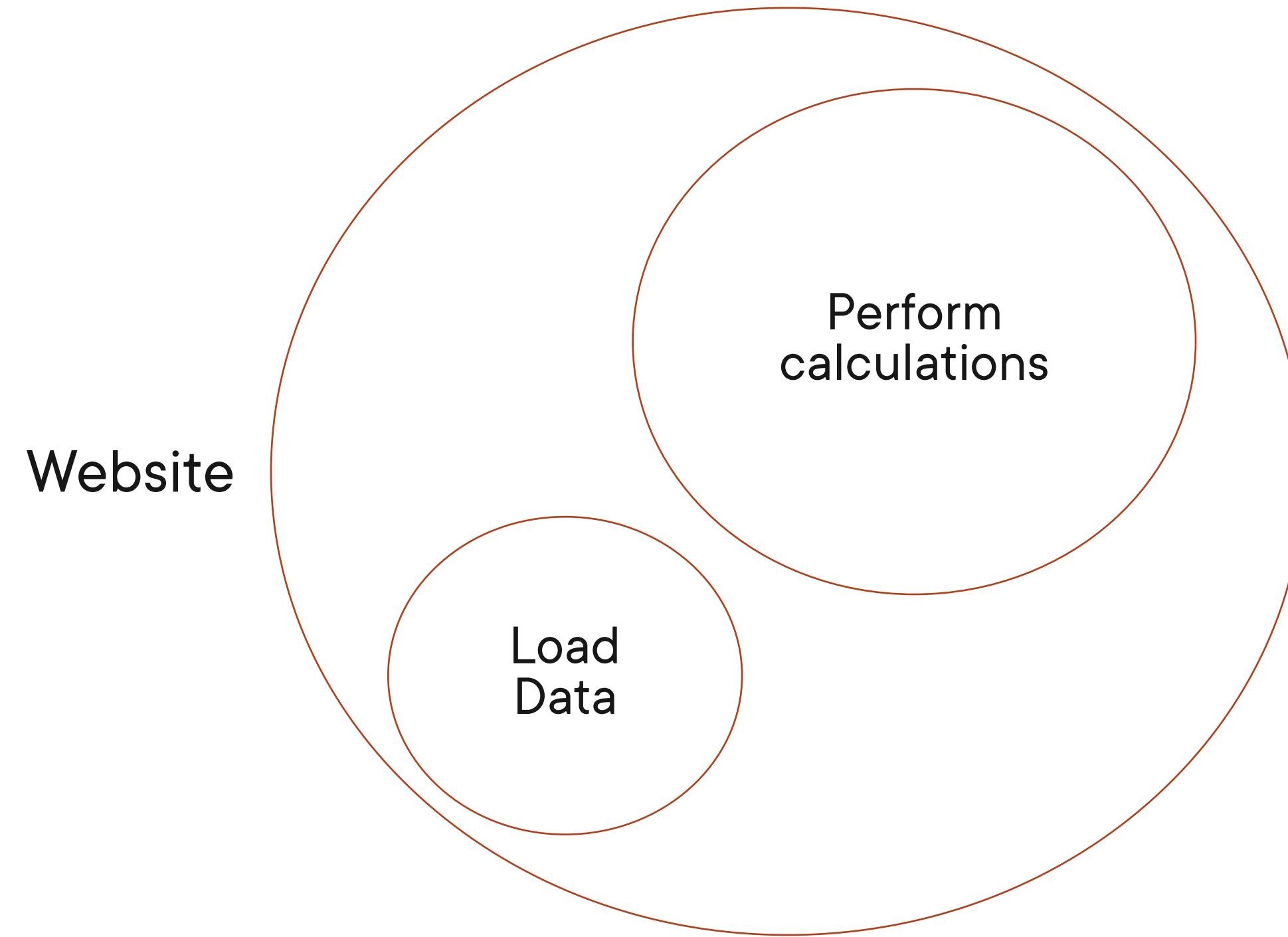
**What if our application is mapping
orbits and doing all that math?**



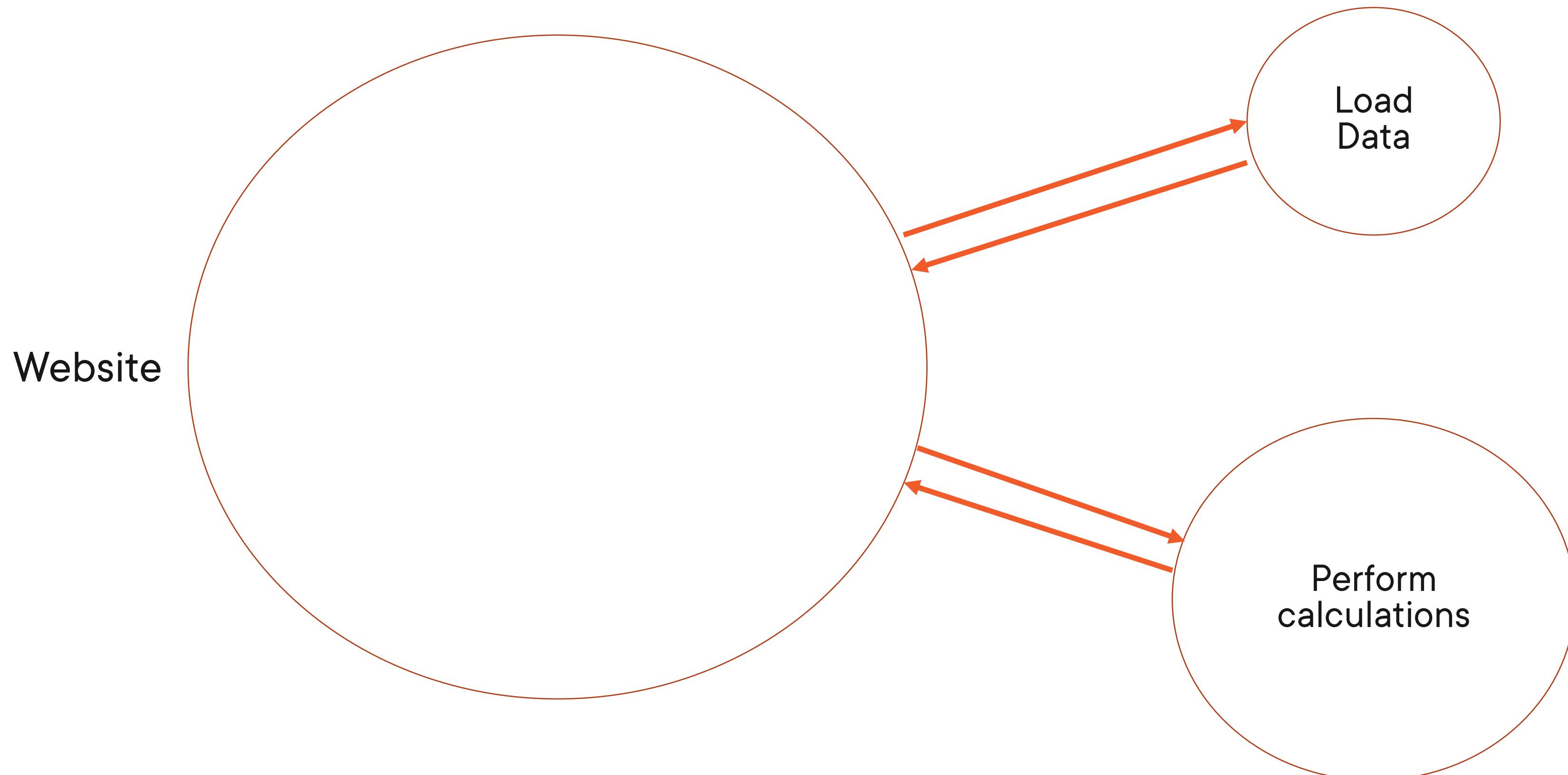
How This Works without DevOps



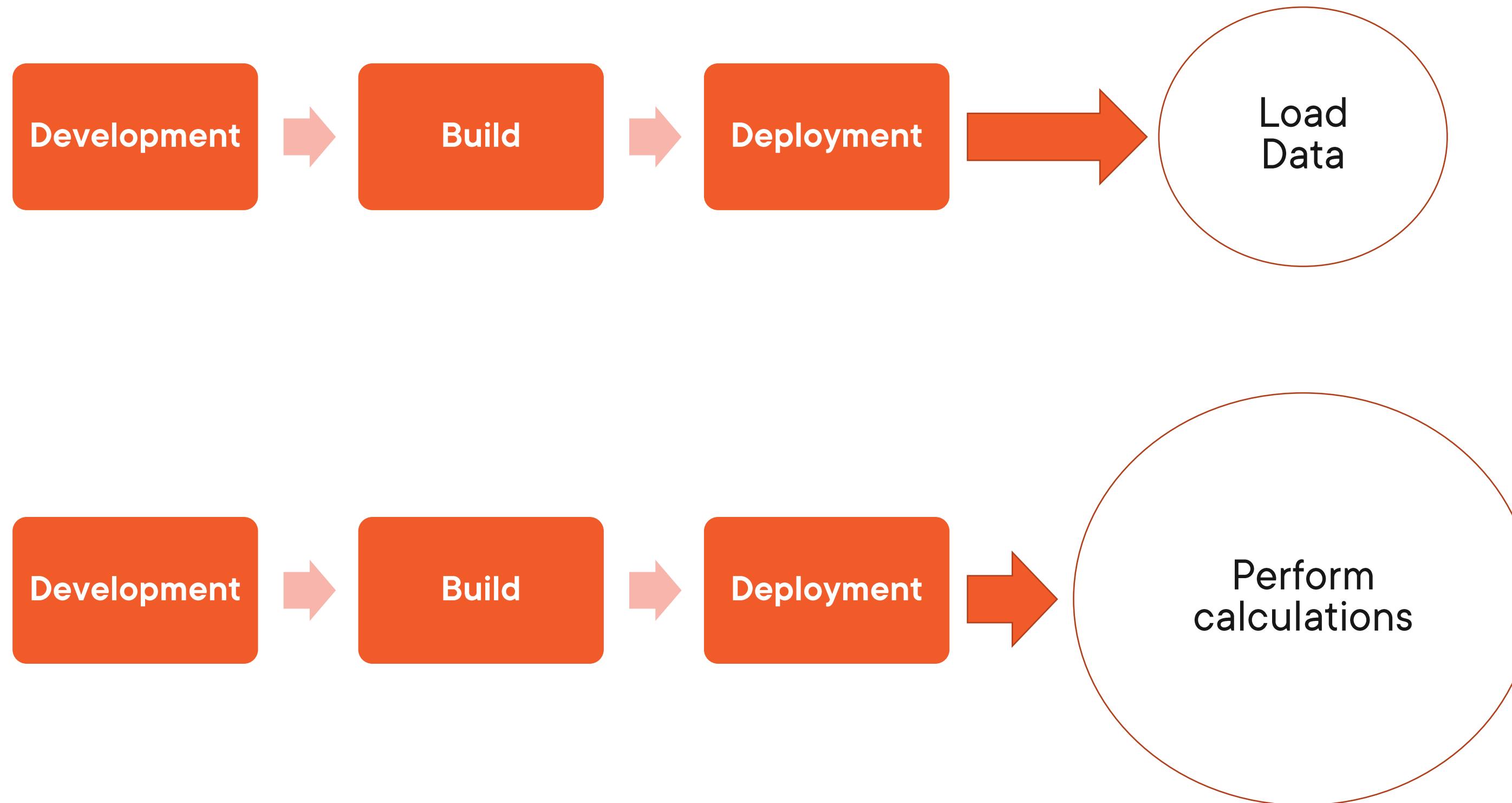
The Dumb Website



The Dumb Website



The Dumb Website



What Else We Get From This

Separation improves testability

**Testability improves
deployability**

**Deploying often improves
everything**

Microservice architecture



In the beginning of DevOps adoption, you will simply be facilitating the existing architecture. To reach the ultimate goal, the architecture will have to change to facilitate DevOps.



Deployment Is the Only Thing



Deployment is always the top priority

“Deployment” is carrying a big load here...

Development architectures optimize for human concerns like ease of use

Prepare yourself and your team for the software architecture to change to facilitate effective DevOps

We haven’t talked about the database schema management yet

We’ll tackle this question in the next section



Infrastructure as Code



Why This Matters



Is this worth the trouble?

Maybe not, at least, not at first

Maybe automating your deployment first makes more sense

Automated infrastructure can be (re)constructed at will

In any case, this is the next problem in our timeline



A script to build your
infrastructure from the
ground up.



What About Docker?

Are containers (Docker or otherwise) IaC?

Yes, because it does what IaC does

Traditional IaC happens on bare metal or a VM

IaC is typically JSON



A Slimmed Down ARM Template

simplearm.json

```
{  
  "resources": [  
    {  
      "type": "Microsoft.Storage/storageAccounts",  
      "apiVersion": "2019-06-01",  
      "name": "[concat('store', uniquestring(resourceGroup().id))]",  
      "location": "[resourceGroup().location]",  
      "kind": "StorageV2",  
      "sku": {  
        "name": "[parameters('storageAccountType')]"  
      }  
    ]  
  }
```

Containers as IaC

Configure a machine that exists

Create a custom execution space



Containers ARE IaC

```
FROM mcr.microsoft.com/windows/servercore:20H2
```

```
RUN powershell -Command `  
Add-WindowsFeature Web-Server; `  
Invoke-WebRequest -UseBasicParsing -Uri  
"https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.10/ServiceMonitor.ex  
e" -OutFile "C:\ServiceMonitor.exe"
```

```
EXPOSE 80
```

```
ENTRYPOINT [ "C:\\ServiceMonitor.exe", "w3svc" ]
```



Configuration Drift

**Configuration
changes after the
initial IaC sync**

**You need to KEEP
your infrastructure
in configuration**

**So, you need a
continual agent**



iis.ps1

```
configuration IIS_Install {
    node localhost {
        WindowsFeature IIS {
            Ensure = "Present"
            Name="Web-Server"
        }
    }
}
```

lis-absent.ps1

```
configuration IIS_NotInstall {
    node localhost {
        WindowsFeature IIS {
            Ensure = "Absent"
            Name="Web-Server"
        }
    }
}
```

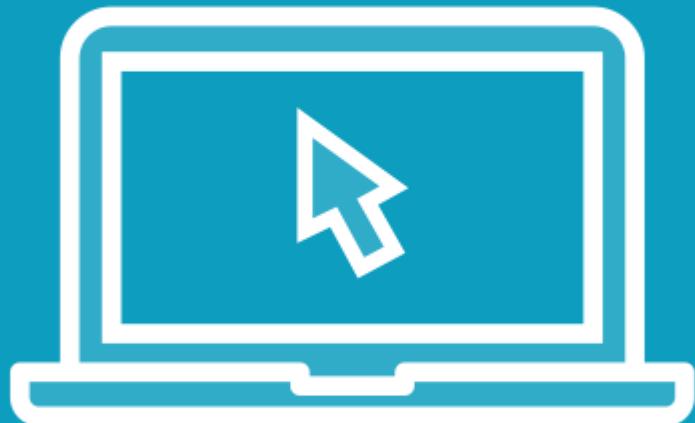
Forging Ahead

We'd probably be focusing on deployment automation...

But let's look at IaC instead



Demo



An Azure Resource Template in Azure

How it works

Running a simple Dockerfile

Talk about how that would work

In the real world



Breadth instead of depth

Specifying Deployment Requirements in Microsoft Azure

<https://bit.ly/3iS24K6>



Dockerfile Wrap-up

**Configure the
server**

**And everything
else that precedes
deployment**

**All this goes in
version control**



Docker Course Links

<https://app.pluralsight.com/library/courses/sql-server-databases-docker-developing>

<https://app.pluralsight.com/library/courses/running-jenkins-docker>

<https://app.pluralsight.com/library/courses/using-microsoft-tye-microservices>



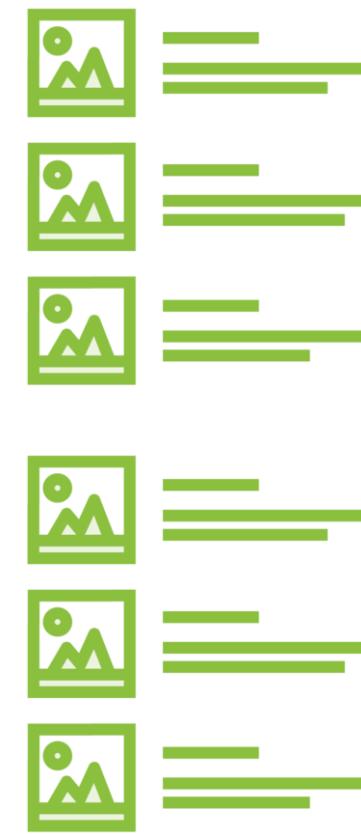
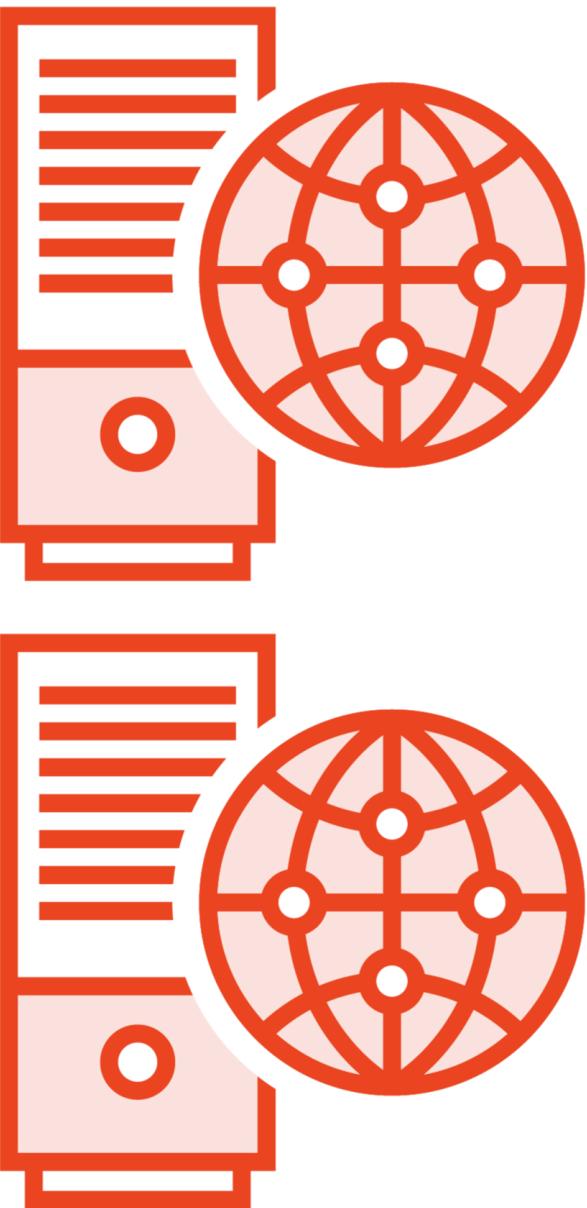
Secrets and Security in DevOps



Everything* belongs in version control.



All Your Stuff



Dynamic Scaling

**When a threshold
of some sort is
exceeded**

**A new resource set
is provisioned**

**But only if your
resources are
organized this way**



Secrets do not belong in
version control.



The Right Way to Do Secrets in DevOps



How do I work with resources that need credentials?

As injected parameters

A Docker file that needs to connect to a private container repo

Specify the creds as environment variables

Then pass the cred from the context to the script

Same for ARM templates

We've only pushed the problem to the context

So, what IS the right place?



What the Right Place Is

A secure store

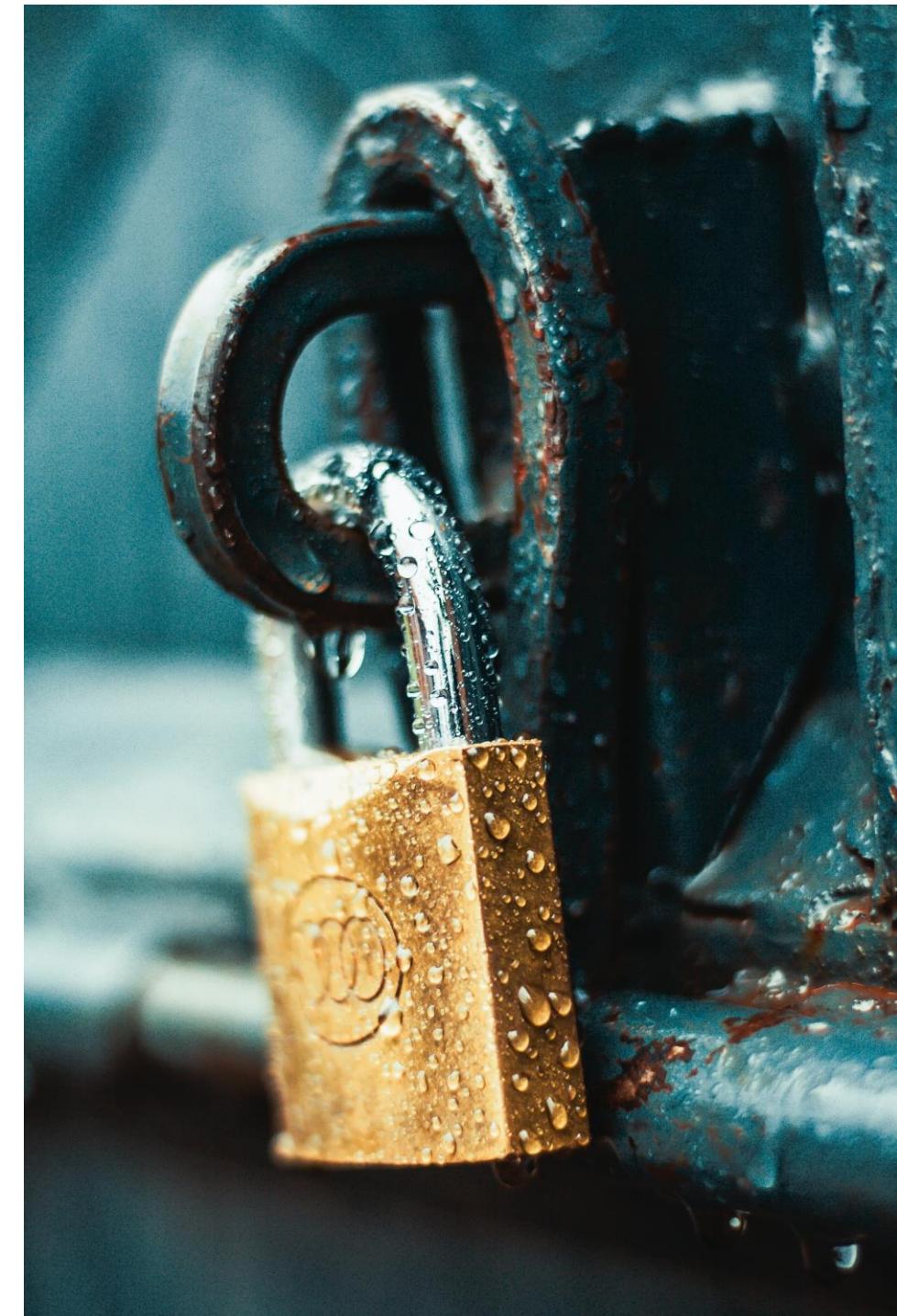
A secure parameter in your build engine

<https://bit.ly/3ggCoFm>

An Azure Key Vault

AWS Secrets Manager

We still haven't really solved the problem



Ultimate Solutions

**Environment
variables in the
execution space**

Set at creation time

**Injected from
secure variables**

**Managed Service
Identity (MSI)**

**The Identity has the
permissions needed**

**All the operations
happen under the
covers**

[https://app.pluralsight.com/library/courses/microsoft-
azure-web-applications-services-deploying](https://app.pluralsight.com/library/courses/microsoft-azure-web-applications-services-deploying)



Summary



Dug into some nuts and bolts

A look at a super-simple Azure pipeline build

Microservice architecture

A look at Infrastructure as Code

Original

Containers

