
Architecture & Infrastructure As Code

About me

Christian Eder

36 years

Father of a 2 years old daughter → just a little bit sleepy today ☺

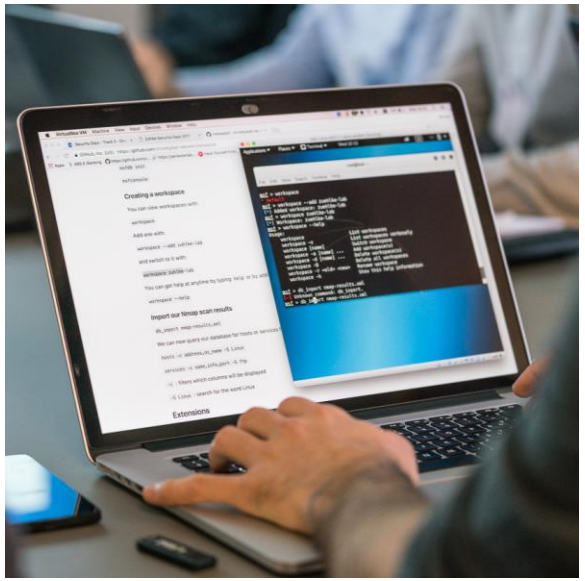
@_ceder

https://github.com/ChristianEder

Our agenda for today

zühlke
empowering ideas

Motivation on the topics covered



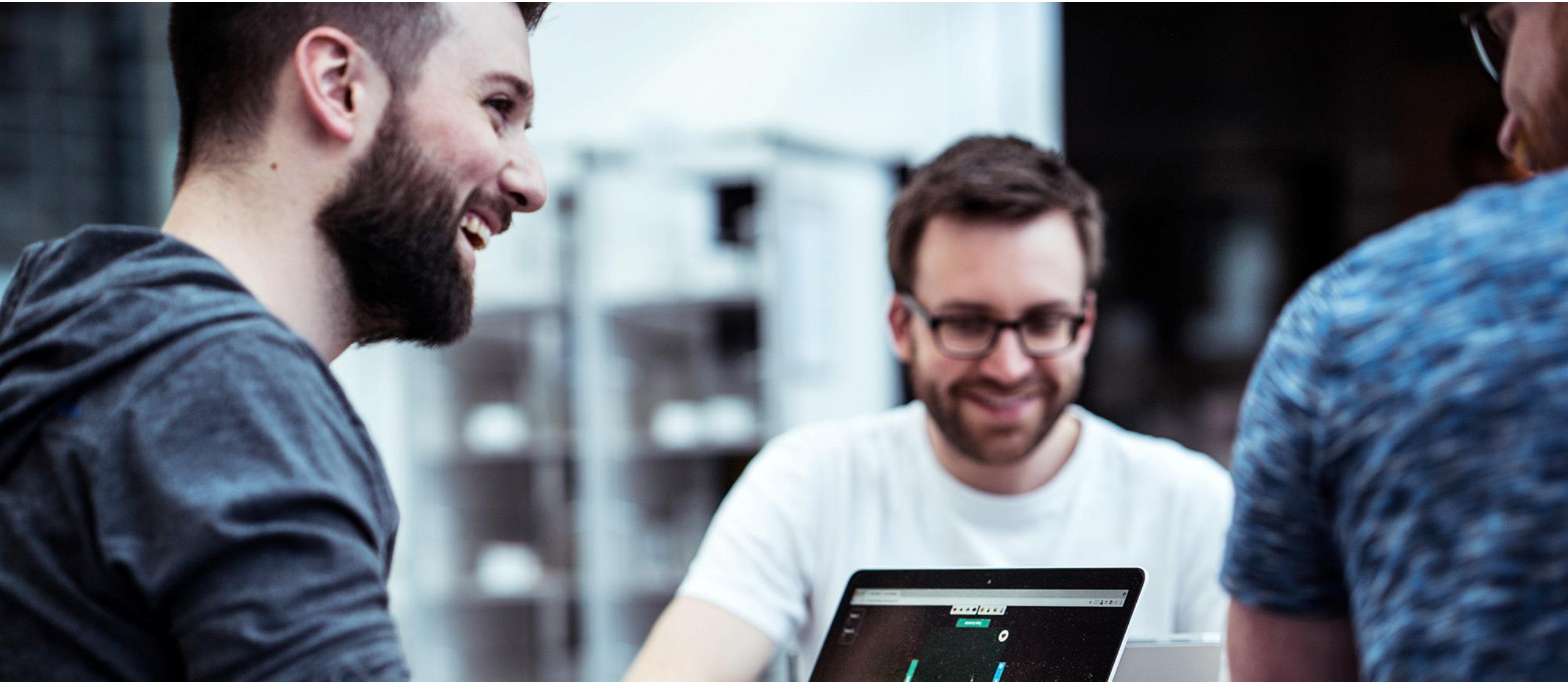
Hands on coding – guided & self paced

Short talks & demos of tool gems

Digging deeper – hands on

About you

zühlke
empowering ideas



Our agenda for today

Motivation on the topics covered

Hands on coding – guided & self paced

Short talks & demos of tool gems

Digging deeper – hands on

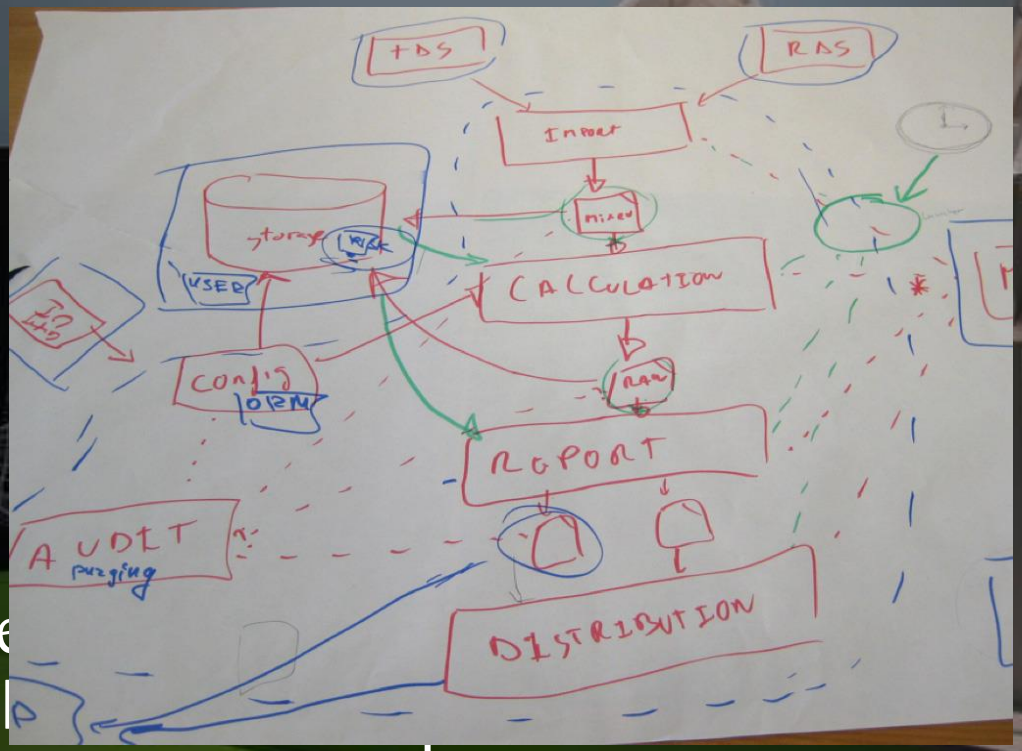
The model code gap

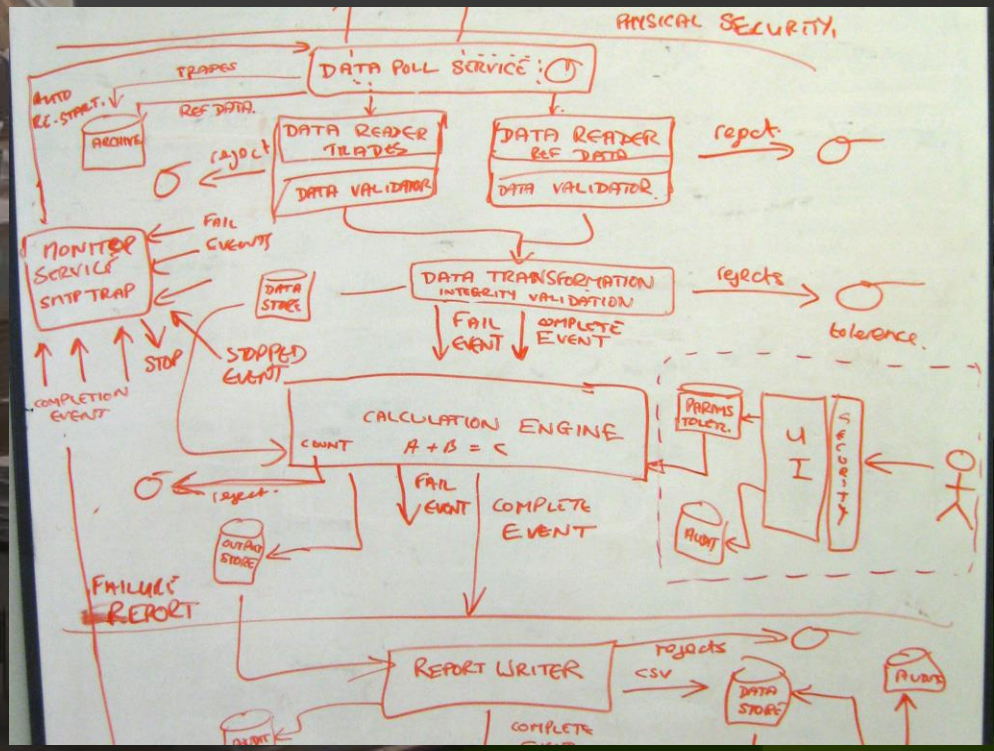
What does this model have in common with most software architecture models?

Most likely will never get updated when the actual thing is being built

Doesn't show a lot technical detail

But at least its beautiful ;-)

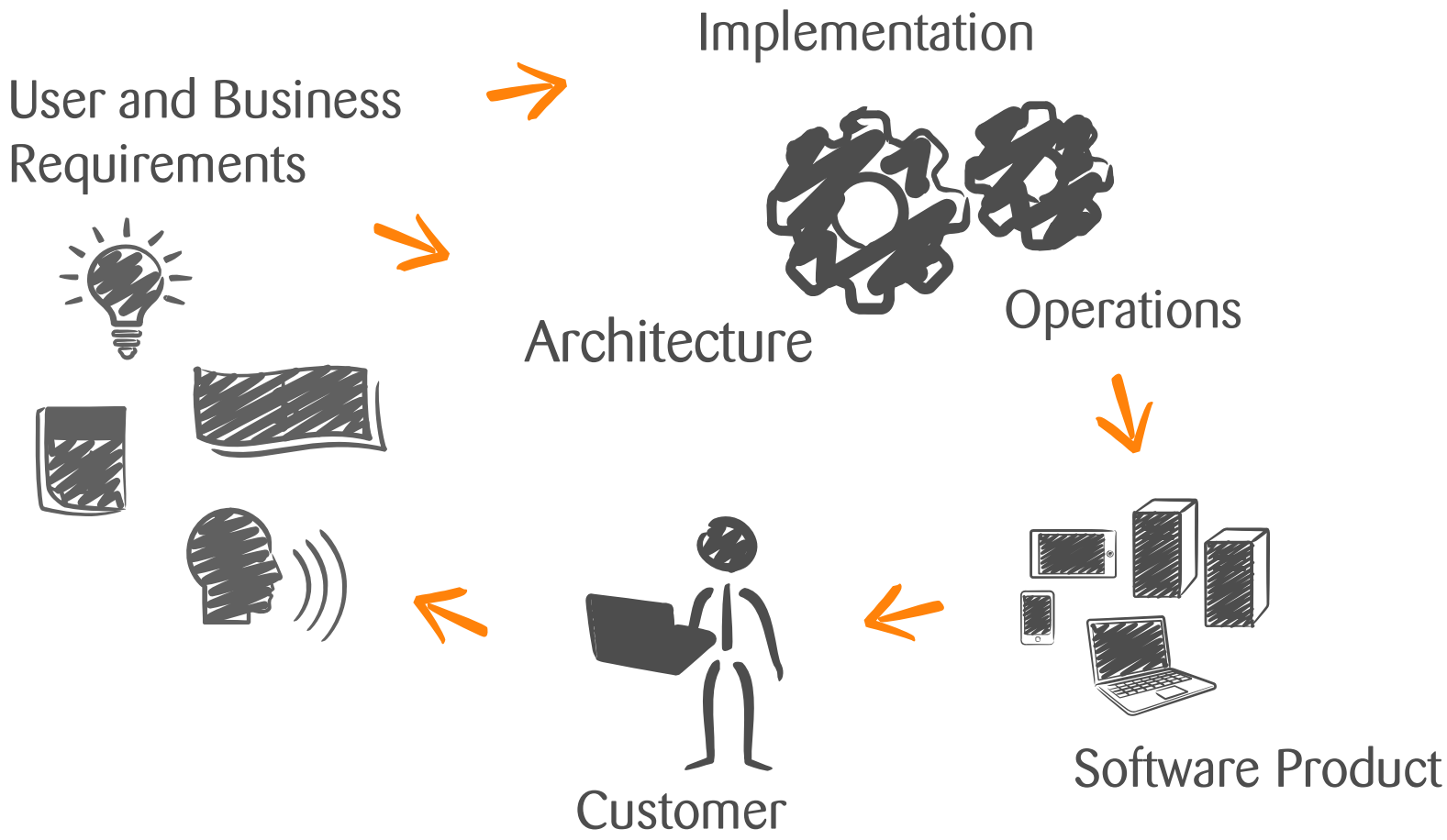




The
will
but not the same

The model code gap

Models are everywhere in software development



The model code gap

When models don't show the same things

Implementation

→ Code

Architecture

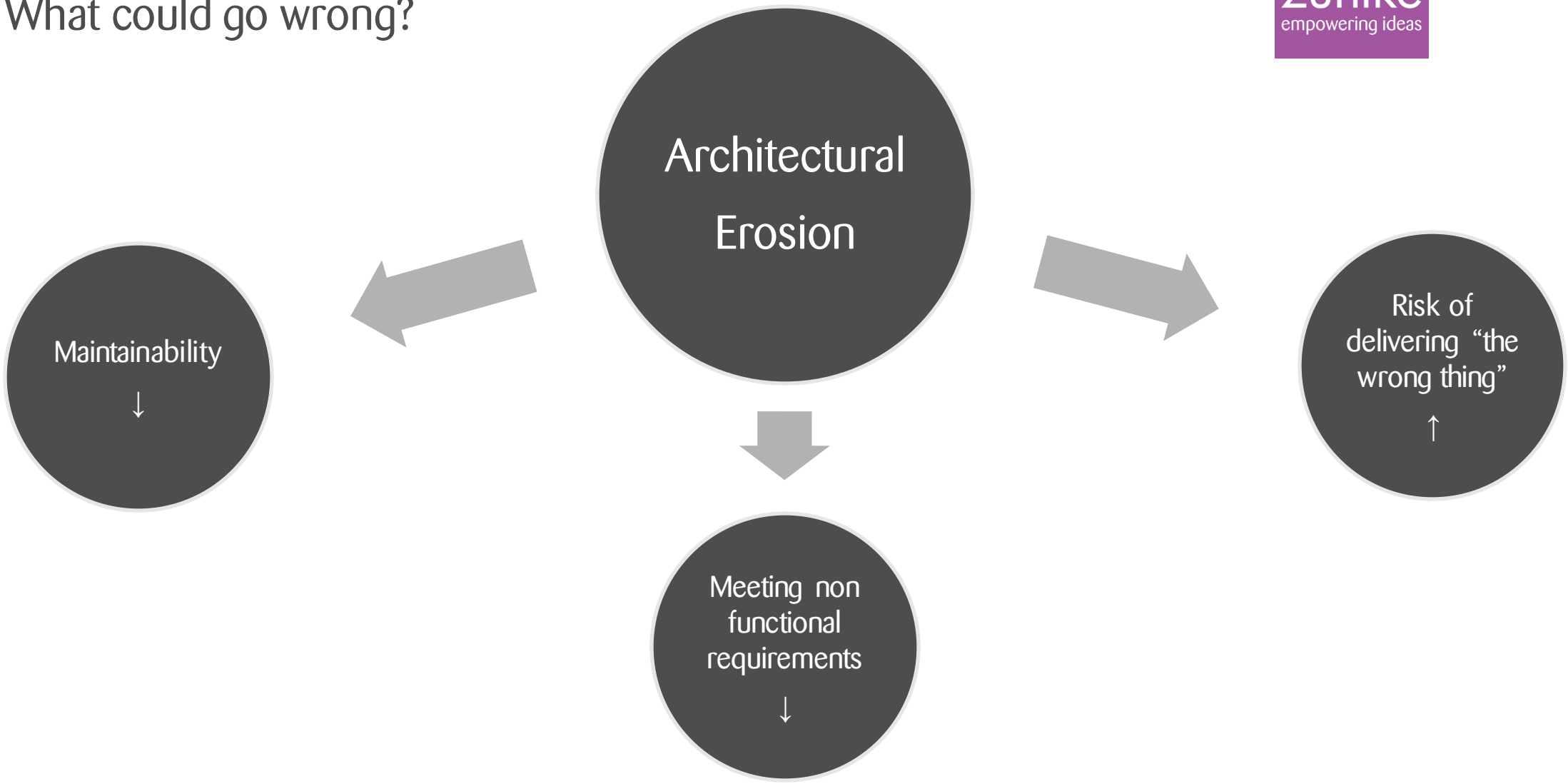
Operations

→ Diagrams & Documents

→ Infrastructure

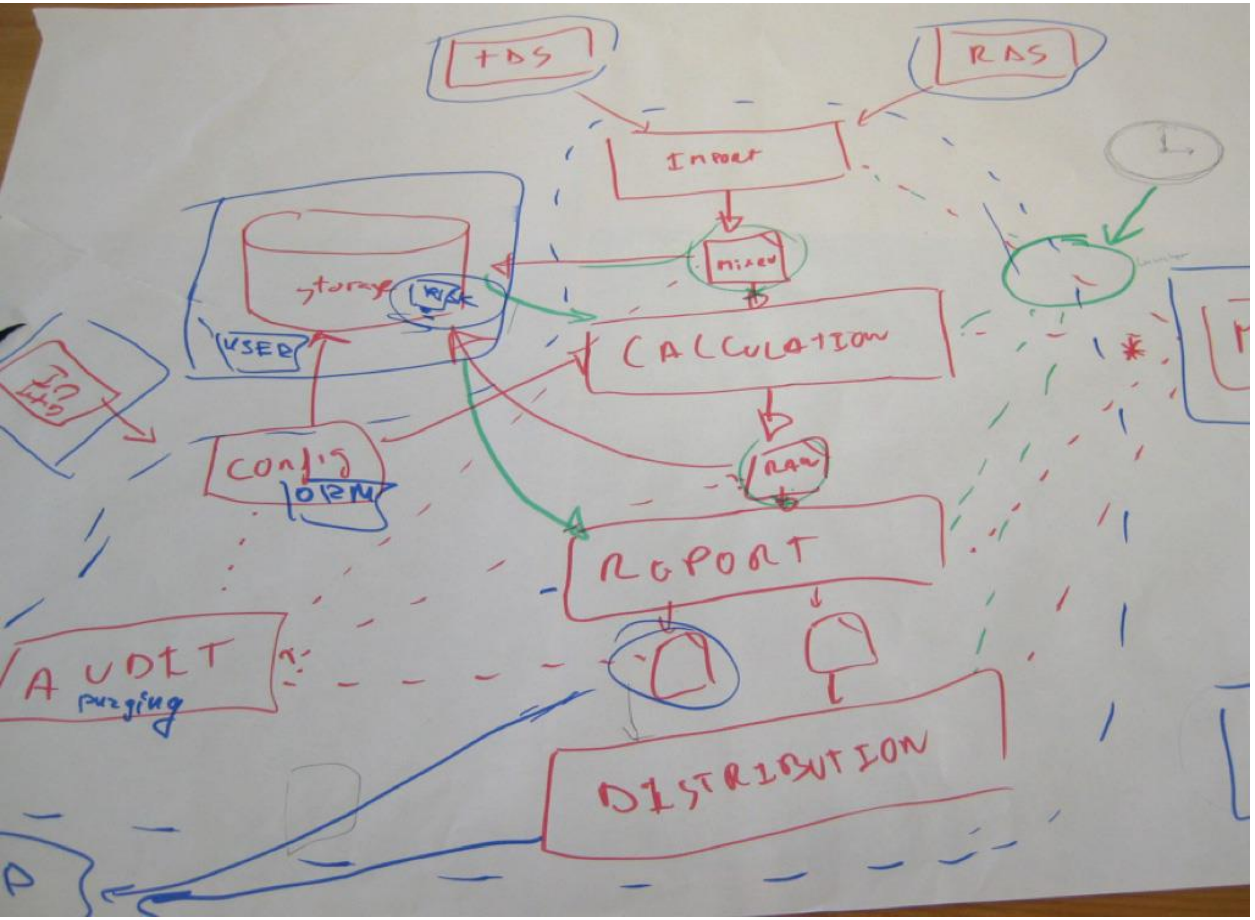
The model code gap - effects

What could go wrong?



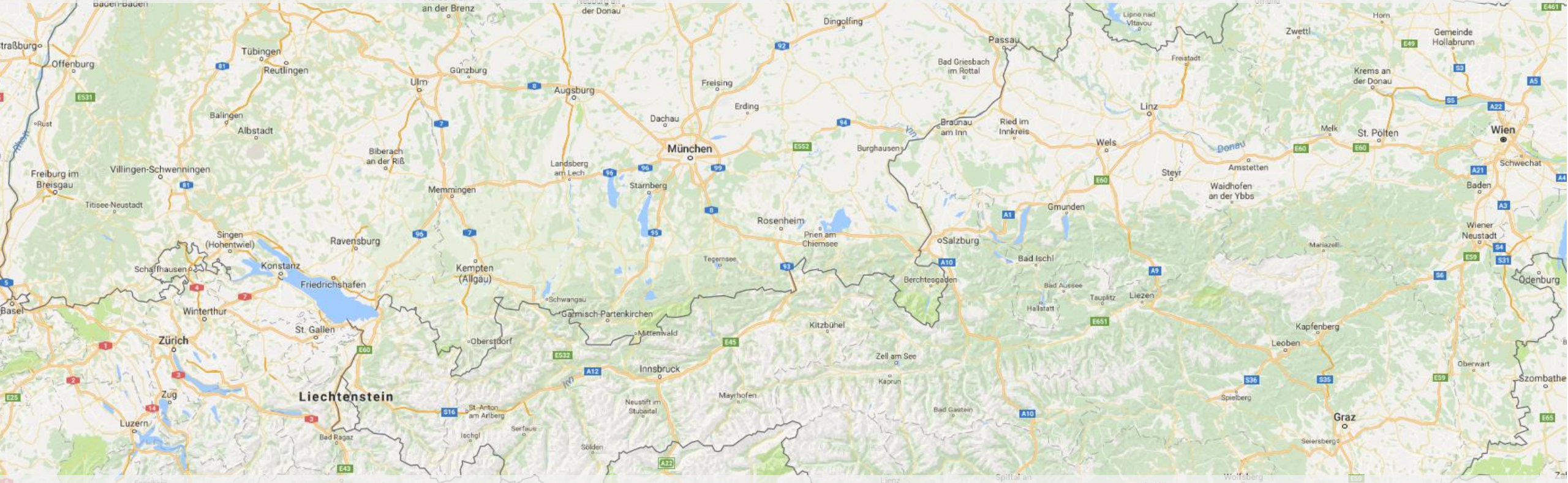
Architecture Model

We all know & fear architecture models like these



Simon Browns C4 Architecture Model

"Diagrams are maps"

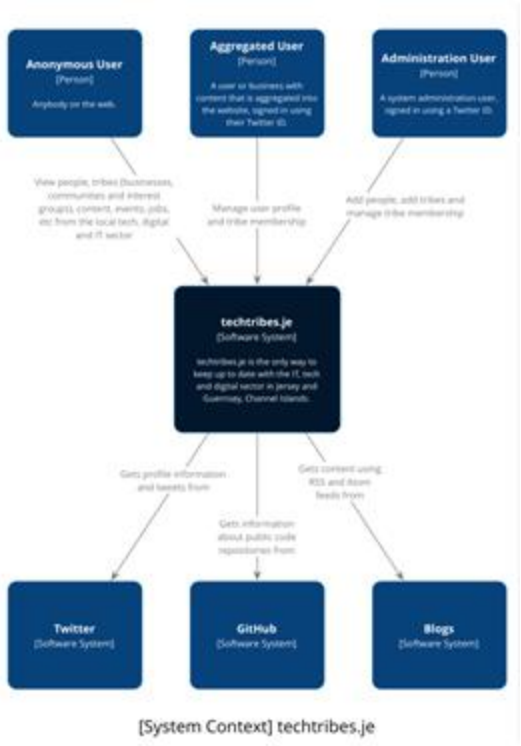


System Context, Containers, Components, Classes

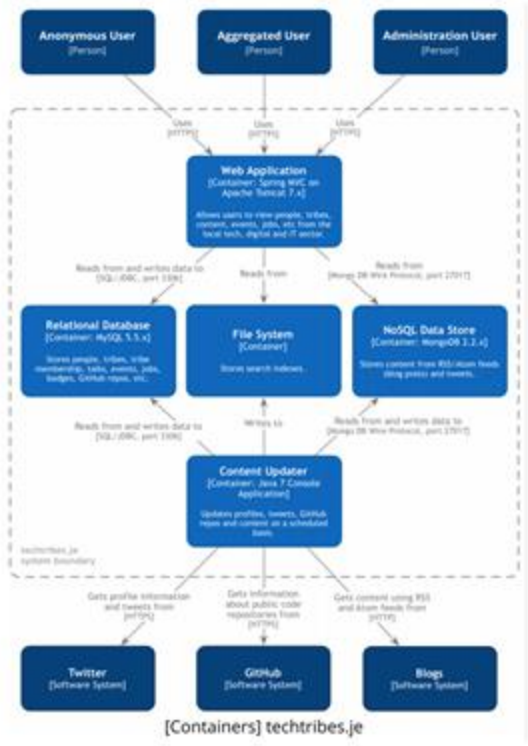
http://simonbrown.je
http://www.codingthearchitecture.com

Simon Browns C4 Architecture Model

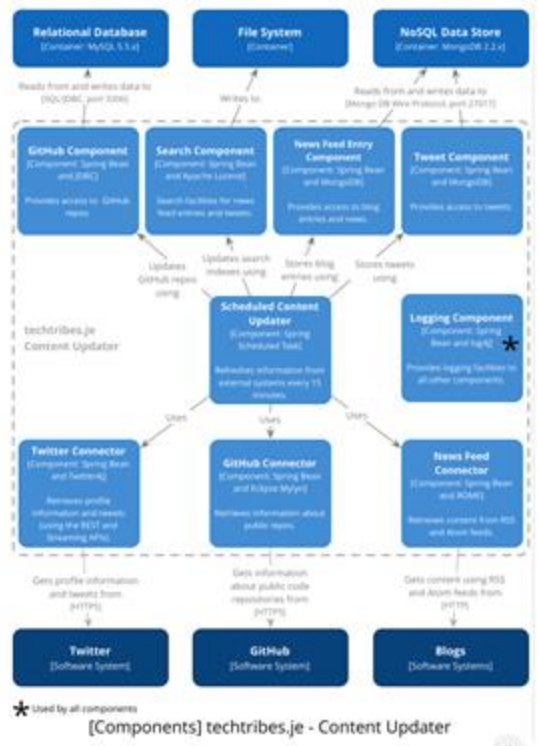
"Diagrams are maps that help you **navigating**"



[System Context] techtribes.je



[Containers] techtribes.je

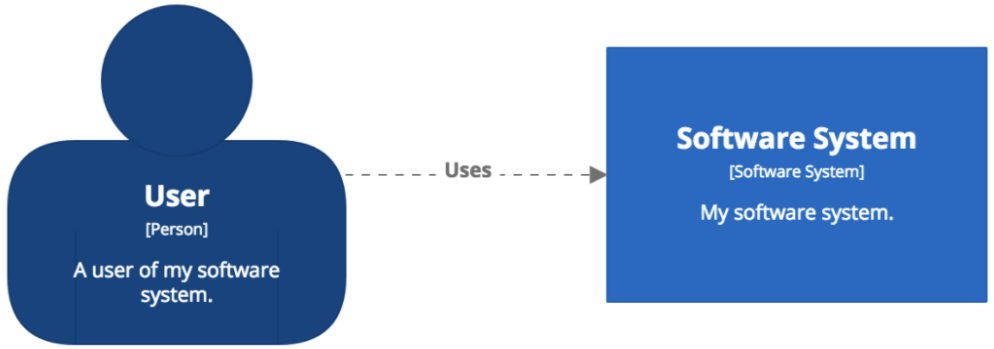


[Components] techtribes.je - Content Updater

Simon Browns Structurizr

Create software architecture models based upon the C4 model using code

```
Workspace workspace = new Workspace("Getting Started", "This is a model of my software system.");
Model model = workspace.getModel();

Person user = model.addPerson("User", "A user of my software system.");
SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
user.uses(softwareSystem, "Uses");

ViewSet views = workspace.getViews();
SystemContextView contextView = views.createSystemContextView(softwareSystem, "SystemContext", "An example of a
contextView.addAllSoftwareSystems();
contextView.addAllPeople();
```

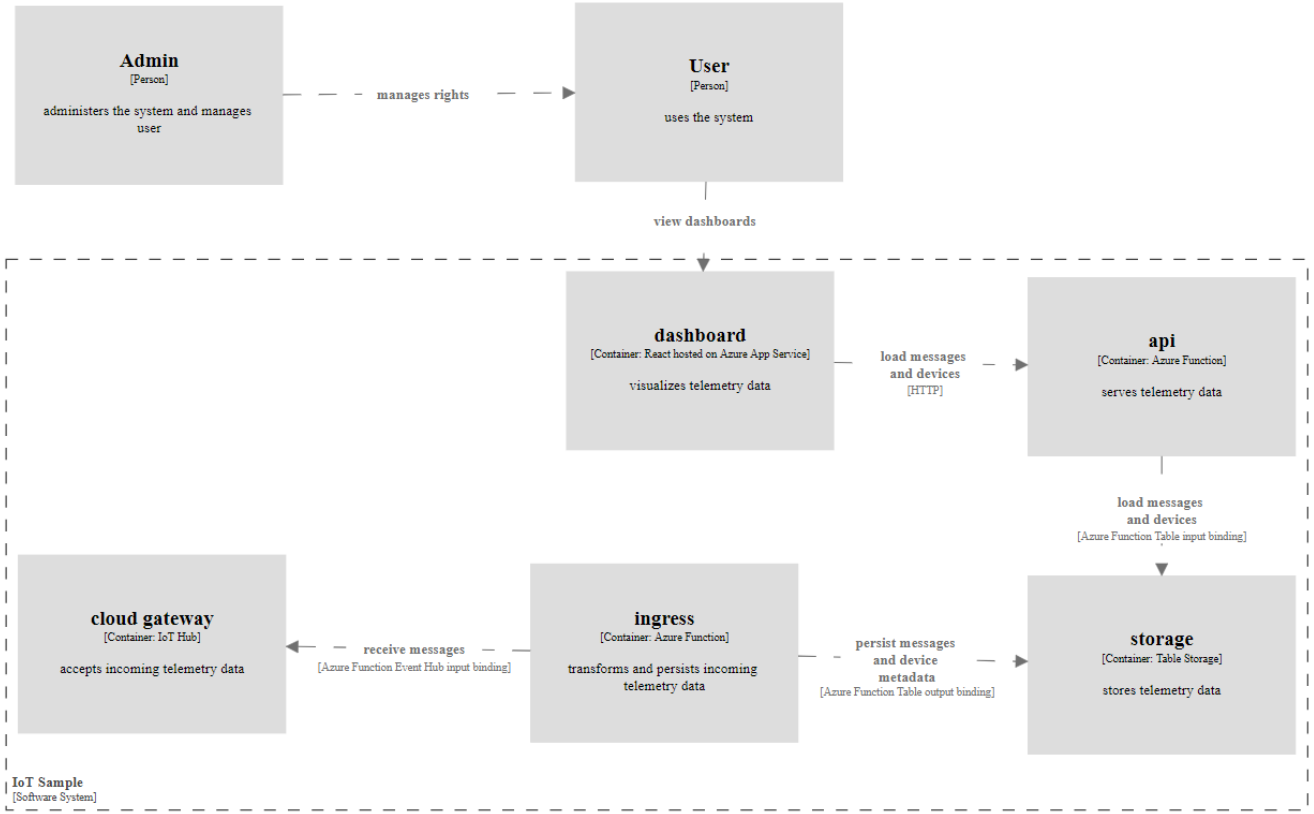


Architecture as *Code*

Using Structurizr – SDKs are available for Java, C# and TypeScript




```
stages:
- stage: Bu
  steps:
  - scrip
    displ
    worki
```



“Infrastructure” as code

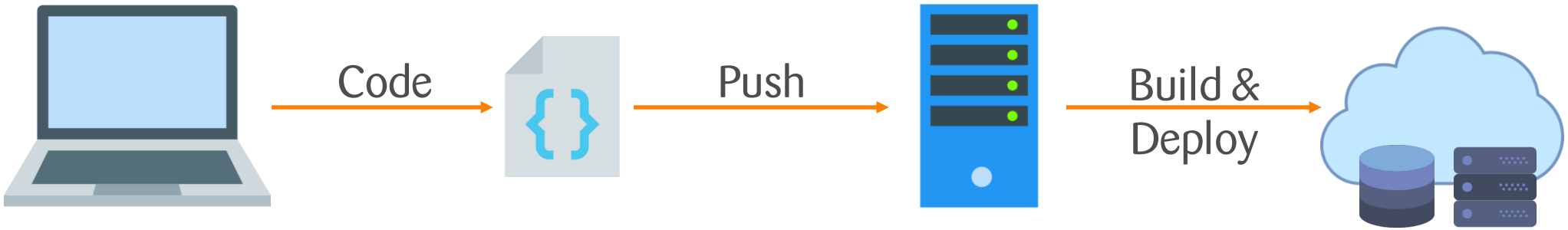
“Infrastructure” ≈ “Something that allows to run code or store data”

IAAS	PAAS	FAAS
Azure Virtual machines AWS EC2	Azure App Services AWS Elastic Beanstalk	Azure Functions AWS Lambda

Azure wording: “Resources”
AWS wording: “Services”
Wording for this talk: “Infrastructure”

Infrastructure as code

Applying software development principles to cloud infrastructure deployment



Infrastructure as code

There are a lot of tools to automate deploying cloud infrastructure

Tools provided by the cloud providers

- **Azure Resource Manager**: JSON files, PowerShell, az CLI
- **AWS Cloud Formation**: YAML files
- **Google Cloud Platform Deployment Manager**: YAML files

Cloud agnostic tools

- **Terraform**: custom DSL with cloud-specific providers
- **pulumi**: TypeScript, Golang, Python

Infrastructure as Code

There are a lot of tools to automate deploying cloud infrastructure

Azure Resource Manager

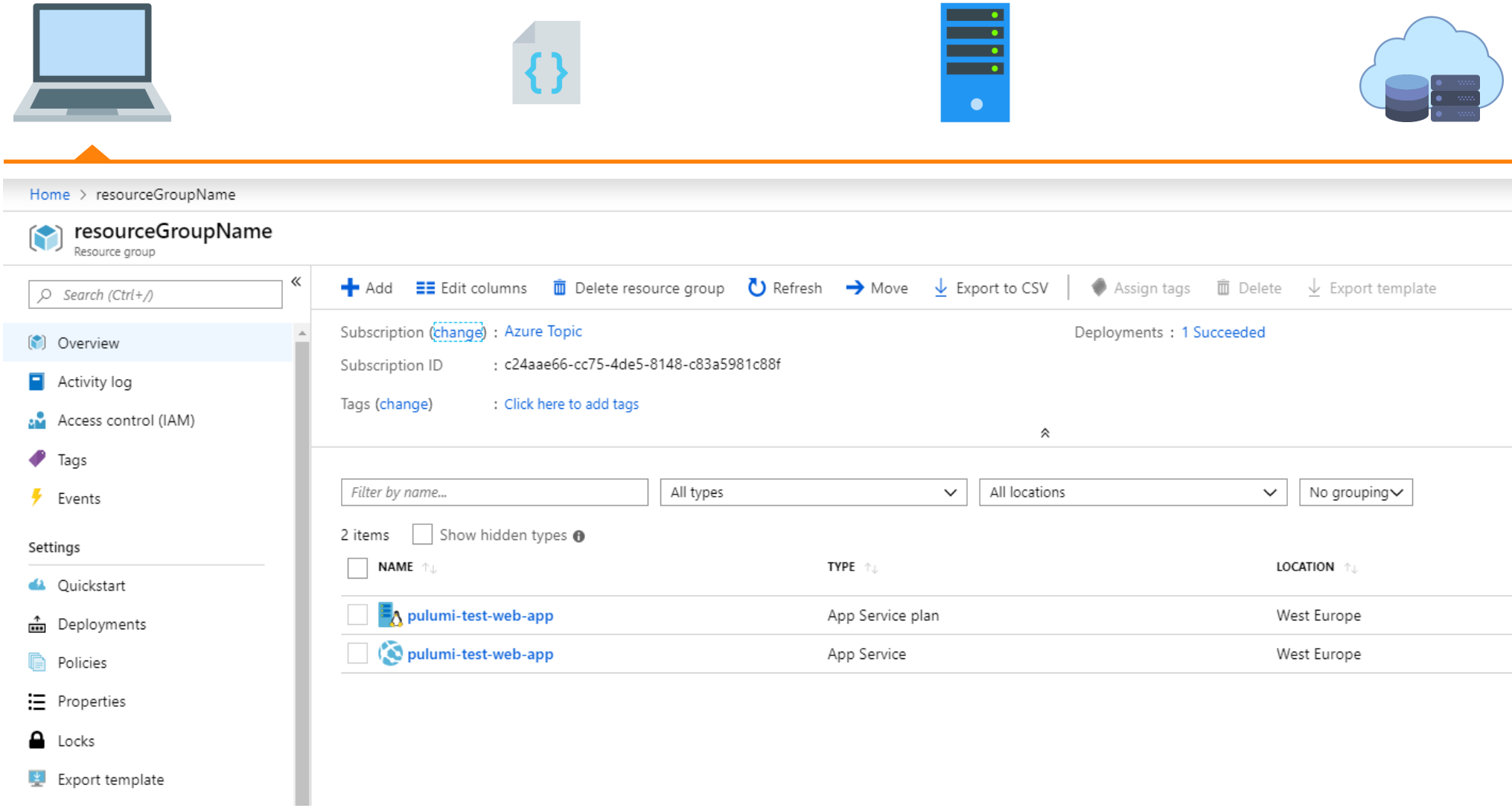
```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "sku": {
            "type": "string"
        },
        "skuCode": {
            "type": "string"
        }
    },
    "variables": {
        "name": "test-web-app"
    },
    "resources": [
        {
            "apiVersion": "2018-02-01",
            "name": "[variables('name')]",
            "type": "Microsoft.Web/sites",
            "location": "[resourceGroup().location]",
            "dependsOn": [
                "[concat('Microsoft.Web/serverfarms/', variables('name')]"
            ],
            "properties": {
                "name": "[variables('name')]",
                "serverFarmId": "[concat(resourceGroup().id, '/providers/Microsoft.Web/serverfarms/', variables('name'))]"
            }
        },
        {
            "apiVersion": "2018-02-01",
            "name": "[variables('name')]",
            "type": "Microsoft.Web/serverfarms",
            "location": "[resourceGroup().location]",
            "kind": "linux",
            "properties": {
                "name": "[variables('name')]",
                "workerSize": "0",
                "workerSizeId": "0",
                "numberOfWorkers": "1"
            },
            "sku": {
                "Tier": "[parameters('sku')]",
                "Name": "[parameters('skuCode')]"
            }
        }
    ]
}
```

Terraform

```
4 references
resource "azurerm_resource_group" "test" {
  name     = "example-resources"
  location = "West Europe"
}

1 references
resource "azurerm_app_service_plan" "test" {
  name                = "example-appserviceplan"
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name = "${azurerm_resource_group.test.name}"

  sku {
    tier = "Standard"
    size = "S1"
  }
}

0 references
resource "azurerm_app_service" "test" {
  name                = "example-app-service"
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name = "${azurerm_resource_group.test.name}"
  app_service_plan_id = "${azurerm_app_service_plan.test.id}"
}
```

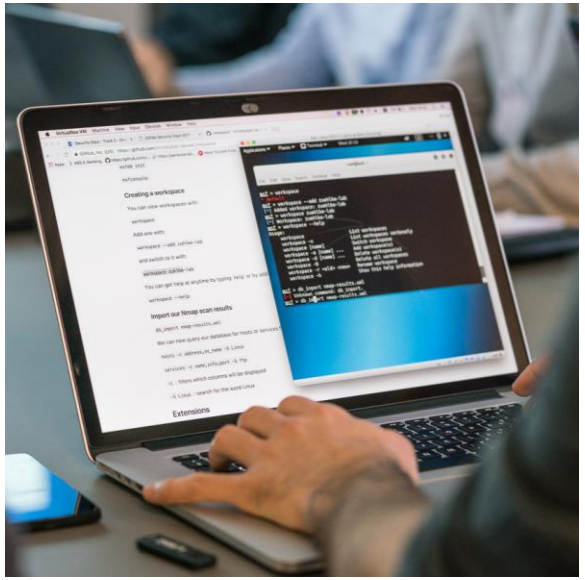
Infrastructure as *Code*

Using pulumi



Our agenda for today

Motivation on the topics covered



Hands on coding – guided & self paced

Short talks & demos of tool gems

Digging deeper – hands on

Hands on

zühlke
empowering ideas

💻 Get the code

git clone https://github.com/ChristianEder/pulumi-structurizr-workshop.git

🛠 Get the tools

- VS Code or Visual Studio https://code.visualstudio.com/download
- .NET Core SDK https://dotnet.microsoft.com/download
  (or NodeJS https://nodejs.org/en/)
- Optional
  - Azure CLI https://docs.microsoft.com/en-us/cli/azure/install-azure-cli
  - Pulumi CLI https://www.pulumi.com/docs/get-started/install/
  - GraphViz & PlantUml (e.g. via Chocolatey)

Hands on

zühlke
empowering ideas

To get started, see readme at

https://github.com/ChristianEder/pulumi-structurizr-workshop.git

Our agenda for today

zühlke
empowering ideas

Motivation on the topics covered

Hands on coding – guided & self paced

Short talks & demos of tool gems

Digging deeper – hands on

Priorization time!

- Testing your infrastructure code

- Enforcing company wide policies for infrastructure code

- Automating code deployments with Pulumi

- Modelling vs. Diagramming – generating multiple views from a single model

- Architecture Decision Records

- Self Contained Components – bringing it all together: Code, Architecture, Infrastructure

Testing Infrastructure Code

https://www.pulumi.com/docs/guides/testing/

zühlke
empowering ideas

- **Unit Tests** are fast in-memory tests that mock all external calls.
- **Property Tests** run resource-level assertions *while* infrastructure is being deployed.
- **Integration Tests** deploy ephemeral infrastructure and run external tests against it.

	Unit Tests	Property Tests	Integration Tests
Provision real infrastructure	No	Yes	Yes
Require the Pulumi CLI	No	Yes	Yes
Time to execute	Milliseconds	Seconds	Minutes
Language	Same as Pulumi program	Node.js or Python	Any language
Validation target	Resource inputs	Resource inputs and outputs	External endpoints

Testing Infrastructure Code

Unit Testing

https://github.com/pulumi/examples/tree/74db62a03d013c2854d2cf933c074ea0a3bbf69d/testing-unit-cs

```
// check 1: Instances have a Name tag.
[Test]
public async Task InstanceHasNameTag()
{
    var resources = await Testing.RunAsync<WebserverStack>();

    var instance = resources.OfType<Instance>().FirstOrDefault();
    instance.Should().NotBeNull("EC2 Instance not found");

    var tags = await instance.Tags.GetValueAsync();
    tags.Should().NotBeNull("Tags are not defined");
    tags.Should().ContainKey("Name");
}
```

Testing Infrastructure Code

Property Testing

zühlke
empowering ideas

Property tests are based on Policy as Code, Pulumi's offering to set compliance for cloud resources.

In addition this enables another type of infrastructure testing.

Each policy becomes a property, an invariant, that a test evaluates and asserts.

```
import * as aws from "@pulumi/aws";
import * as policy from "@pulumi/policy";
import * as pulumi from "@pulumi/pulumi";

const stackPolicy: policy.StackValidationPolicy = {
    name: "eks-test",
    description: "EKS integration tests.",
    enforcementLevel: "mandatory",
    validateStack: async (args, reportViolation) => {
        const clusterResources = args.resources.filter(r => r.isType(aws.eks.Cluster));
        if (clusterResources.length !== 1) {
            reportViolation(`Expected one EKS Cluster but found ${clusterResources.length}`);
            return;
        }
    },
}
const tests = new policy.PolicyPack("tests-pack", {
    policies: [stackPolicy],
});
```

Testing Infrastructure Code

Integration Testing

https://github.com/pulumi/examples/tree/05ae8e1803d7f44cecac69589175e416e421cdfe/testing-integration

*Integration testing takes a different approach of unit tests: the tests deploy cloud resources and validate their actual **behavior**.*

Enforcing Infrastructure Policies

https://www.pulumi.com/docs/guides/crossguard/
https://github.com/pulumi/examples/blob/master/policy-packs/azure-ts/index.ts

zühlke
empowering ideas

CrossGuard empowers you to set ***guardrails*** to enforce ***compliance*** for resources

so ***developers within an organization*** can provision their own infrastructure

while ***sticking to best practices*** and ***security compliance***.

Enforcing Infrastructure Policies

https://www.pulumi.com/docs/guides/crossguard/
https://github.com/pulumi/examples/blob/master/policy-packs/azure-ts/index.ts

```
const policies = new PolicyPack("azure", {
    policies: [
        {
            name: "discouraged-public-ip-address",
            description: "Associating public IP addresses is discouraged.",
            enforcementLevel: "advisory",
            validateResource: validateResourceOfType(azure.network.NetworkInterface, (ni, args, reportViolation) => {
                if (ni.ipConfigurations.some(cfg => cfg.publicIpAddressId)) {
                    reportViolation("Associating public IP addresses is discouraged.");
                }
            }),
        },
```

Automating Code Deployment

https://github.com/pulumi/pulumi-azure/blob/master/examples/multi-callback-all/index.ts
https://dev.to/_ceder/deploying-azure-functions-using-pulumi-net-3dcb
https://www.nuget.org/packages/Scratchpad.NET.Azure.Functions/

```
class Program
{
    static Task<int> Main() => Deployment.RunAsync<AzureFunctionsStack>();
}

2 references
public class AzureFunctionsStack : Stack
{
    public AzureFunctionsStack()
    {
        var resourceGroup = new ResourceGroup("myapp", new ResourceGroupArgs  { Location = "WestEurope" });
        var functionApp = new CallbackFunctionApp("myapp", new CallbackFunctionAppArgs
        {
            ResourceGroupName = resourceGroup.Name,
            Functions = CallbackFunction.FromAssembly(GetType().Assembly)
        });
    }
}
```

Modelling vs. Diagramming

https://structurizr.com/help/modelling
https://structurizr.com/share/39593/985c2c39-fd88-4e2b-91e0-6c93c73b0a84

zühlke
empowering ideas

When ***diagramming***, you're creating separate diagrams, often with an ad hoc notation, using tools that don't understand anything about the ***semantics*** of your diagrams.

With ***modelling***, you're building up a ***non-visual model*** of something, and then ***creating different views*** (e.g. diagrams) on top of that model.

Architecture Decision Records

https://structurizr.com/help/decision-log

zühlke
empowering ideas

Diagrams alone can't express the ***decisions that led to a solution***, Structurizr allows you to supplement your software architecture model with a ***decision log***

2017

8. Use ISO 8601 Format for Dates Tuesday, 21 February 2017	Accepted

2016

7. Invoke adr-config executable to get configuration Saturday, 17 December 2016	Accepted
6. Packaging and distribution in other version control repositories Tuesday, 16 February 2016	Accepted
5. Help comments Saturday, 13 February 2016	Accepted
4. Markdown format	Accepted

Architecture Decision Records

https://structurizr.com/help/decision-log

zühlke
empowering ideas

Add ADRs directly from code…

```
workspace.documentation.addDecision(factory, '1', new Date('2008-09-15T15:53:00'), 'Use ISO 8601 Format for Dates', DecisionStatus.Accepted, Format.Markdown, `W
workspace.documentation.addDecision(undefined, '2', new Date('2008-09-15T15:53:00'), 'Use angular as the frontend framework', DecisionStatus.Proposed, Format.Ma
```

… or import existing markdown files

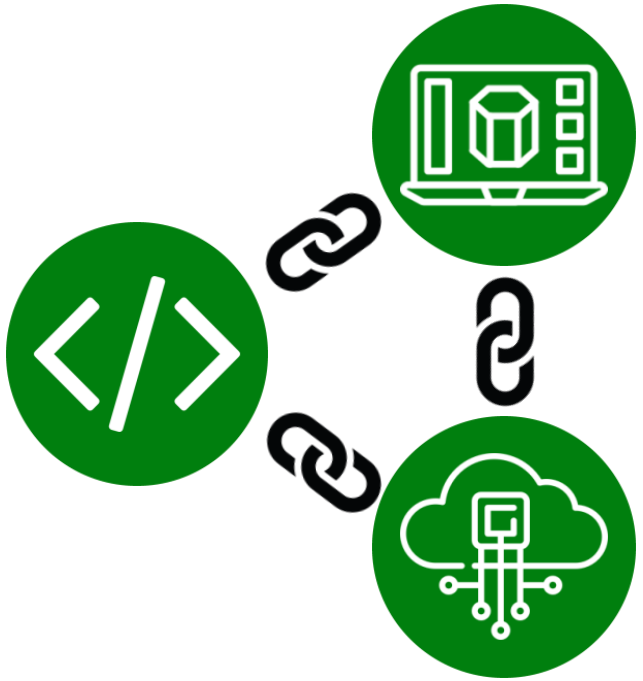
```
AdrToolsImporter adrToolsImporter = new AdrToolsImporter(workspace, adrDirectory);
adrToolsImporter.ImportArchitectureDecisionRecords(adrTools);
```

Bringing It All Together — Self Contained Components

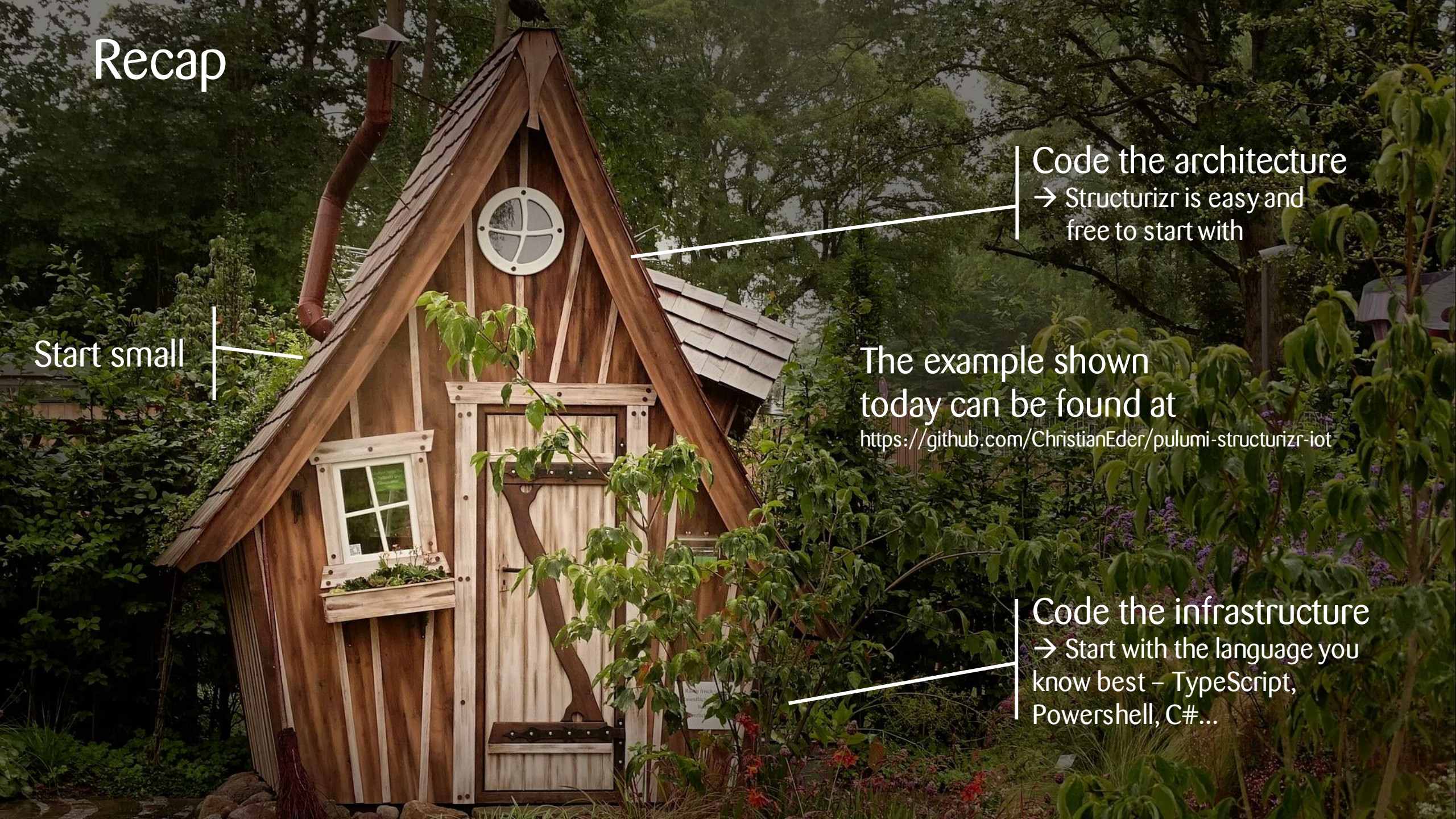
https://github.com/ChristianEder/pulumi-structurizr-workshop/tree/master/labs/dotnet/pulumi-and-structurizr

Closing the model-code gap by implementing our system as a set of components that come with

- A ***model*** of their own ***internal architecture*** as well as ***interfaces*** with other components
- The ***code*** implementing this architectural model
- The ***infrastructure*** required to run that code



Recap

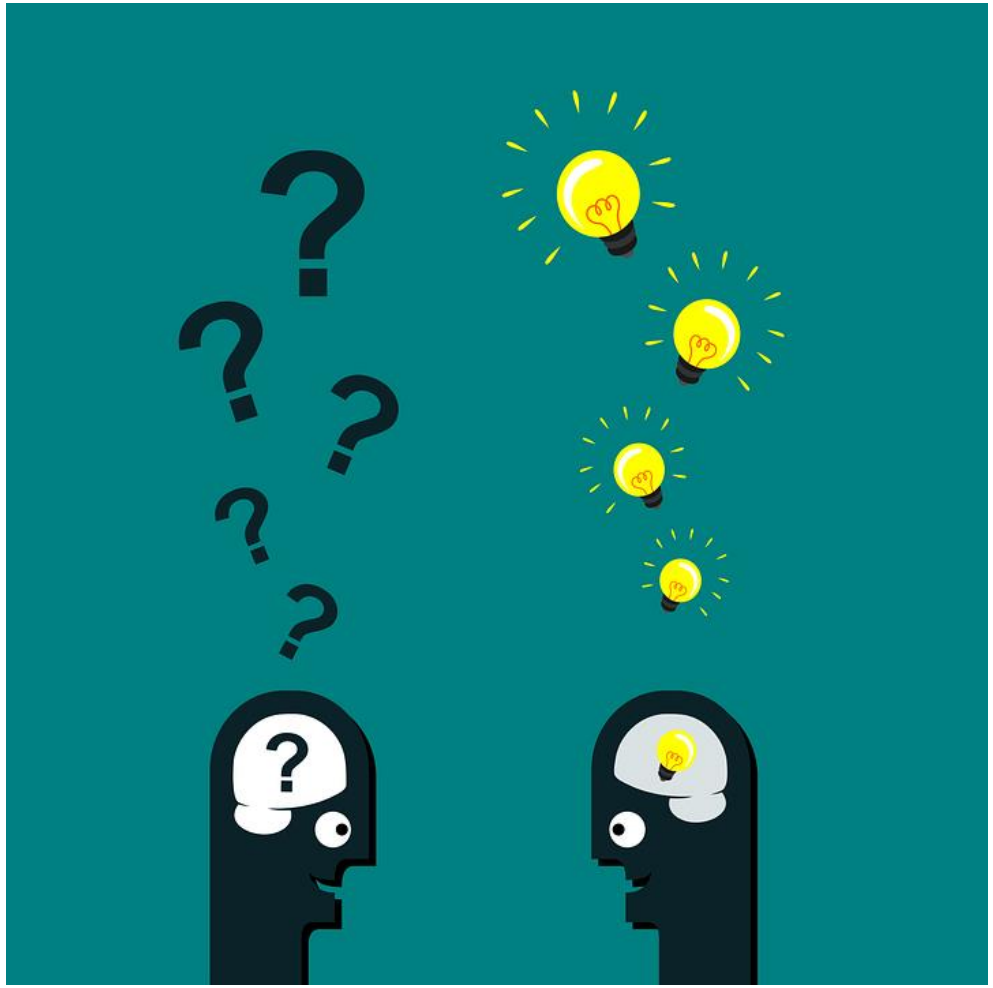


Code the architecture
→ Structurizr is easy and free to start with

Start small

The example shown today can be found at
https://github.com/ChristianEder/pulumi-structurizr-iot

Code the infrastructure
→ Start with the language you know best – TypeScript, Powershell, C#...



zühlke
empowering ideas

@_ceder

https://github.com/ChristianEder

Thank you for listening – any questions?