FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Towards DevOps: Practices and Patterns from the Portuguese Startup Scene

Carlos Manuel da Costa Martins Teixeira

DISSERTATION PLANNING

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Hugo Sereno Ferreira

Co-Supervisor: Tiago Boldt Sousa

May 18, 2016

Towards DevOps:
Practices and Patterns from the Portuguese Startup Scene

Carlos Manuel da Costa Martins Teixeira

Mestrado Integrado em Engenharia Informática e Computação

May 18, 2016

Contents

Chapter 1

Patterns

Acquiring Computing Resources

Entities usually have the need to have a **web** presence or to acquire computing resources to do some tipe of processing. This means that computing resources must be acquired.

Cloud

Context

You have a piece of software that you want to make available for people to interact with.

Because of its size, complexity, distributed nature, etc, you do not want people to have to install that piece of software on their devices.

In order to allow people to interact with your software you have therefore to have some underlying infrastructure to support your needs.

Problem

How do setup/acquire infrastructure/resources in order to meet your needs ?

Forces

- Hardware is expensive to buy and you may only need it for a short period.
- Hiring a team/person for managing and maintaining your hardware is expensive.
- In the future resource needs may vary and you may want to increase or decrease your computational resources.

Solution

Cloud providers allow individuals and businesses to purchase computing resources with a **pay-as-you-go** business model. Cloud providers assume the responsabilitie of **maintaining** and **managing** the physical infrastructure.

Rationale

Due to its scale cloud providers can usually provide a competitive price compared to the price of maintaing you own infrastructure. The pay-as-you-go model allows you to allocate and deallocate resources meaning that you are only using what you need. Together with the fact that no hardware upfront costs are required this means that using the cloud is usually the best alternative.

Platform as a Service

Context

You have decided to use the *Cloud*.

The software you want to deploy follows common flows of installation/setup and you do not need a lot of access to the underlying infrastructure.

Problem

Cloud providers offer a set of different types of services and you do not know which to use ?

Forces

- Platform as a Service usually allow for some configuration of the underlying layer.
- Platform as a Service environments are usually managed by the provider.
- Platform as a Service is usually expensier than Infrastructure as a Service.

Solution

Deploy your software by using a Platform as a Service.

Rationale

PaaS pushes the environment management to the provider. This means that you do not have to alocate people and time to manage it. This does not mean however that there is no control, some configuration can still be specified by the user.

Infrastructure as a Service

Context

You have decided to use the *Cloud* and your application needs **very specif configurations** to be deployed or you need access to the undelying infrastructure.

Problem

What aaS model should you use?

Forces

- Using Infrastructure as a Service means having the responsability of configuring and maintainig all machines.
- Pricewise Infrastructure as a Service is usually the cheapest of the aaS.

Solution

Deploy your software by using IaaS

Rationale

IaaS gives fine grain control over the aspects of the infrastructure. Cloud providers allow for the allocation of virtual machines that are fully configurable.

Reproducible Environments

Containers

Context

You have or want to have several instances of your software installed in several machines. Your instances may need to be updated/altered both in terms of the software they run but also the environment on wich it runs.

Problem

How do you make sure that the environments were your code run are consistent across all machines even when changes are applied ?

Forces

- Performing manual installations can lead to errors.
- Manually installing or updating dependencies and configuring machines is time consuming.
- If changes are costly there will be an incentive not perform them.

Solution

Containerize your applications by creating a container that includes among other things all of your dependencies and configurations.

Virtual Machine

Context

You have or want to have several instances of your software installed in several machines. Your instances may need to be updated/altered both in terms of the software they run but also the environment on wich it runs.

Problem

How do you make sure that the environments were your code run are consistent across all machines and that changes to that environment can be easily applied ?

Forces

- Performing manual installations can lead to errors.
- Manually installing or updating dependencies and configuring machines is time consuming.
- If changes are costly there will be an incentive not perform them.

Solution

Create a **image** of the machine desired.
That machine image can then be copied and used to setup a reproducible environment.

Scripting

Context

ou have or want to have several instances of your software installed in several machines. Your instances may need to be updated/altered both in terms of the software they run but also the environment on wich it runs.

Problem

How do you make sure that the environments were your code run are consistent across all machines and that changes to that environment can be easily applied ?

Forces

- Performing manual installations can lead to errors.
- Manually installing or updating dependencies and configuring machines is time consuming.
- If changes are costly there will be an incentive not perform them.

Solution

Create a script that describes your infrastructure dependencies and configurations. This script can then be used in several machines to create similar environments.

Creating a new instance

Download the Virtual Machine image

Context

You have choose to define your infrastructure using a *Virtual Machine* and you want to create a new runnning instance of your software.

Problem

How do you do it ?

Forces

- Rebuilding the image takes time and depending on how you setup the image build the final image may be different across builds.
- Some of your dependencies may need to be downloaded from external providers. If they fail usually your build fails.

Solution

Build the image you want to run once. Then make it available for download and reuse it.

Pull the container

Context

You have choose to define your infrastructure using a *Containers* and you want to create a new runnning instance of your software.

Problem

How do you do it ?

Forces

- Rebuilding a container takes time and depending on how you setup the container build the final container may be different across builds.

- Some of your dependencies may need to be downloaded from external providers. If they fail usually your build fails.
- If the environment setup has some complex or time-consuming steps you may take a long time to build single a container.

Solution

Build the container once. Then make it available for download and reuse it.

Anti: Run the script

Context

You have choose to define your infrastructure using only *Scripts* and you want to create a new runnning instance of your software.

Problem

How do you do it ?

Forces

- Some of your dependencies may need to be downloaded from external providers. If they fail usually your script fails.
- Re running the script takes time and depending on how you setup the script inconsistencies can arise.

Solution

Run the script that does the configuration of the application environment in the machine were you want to install the application.

Scalling Infrastructure

Anti: Vertical Scalling

Context

You have variable usage of your infrastructural resources. You may need to quicly increase or decrease the size of the infrastructure to meet your needs.

Problem

How do you scale your computing capacity in order to respond to the change.

Forces

- Due to physical constraints there is a limit for how many resources you can alocate to a single machine.
- Increasing the resources of one machine may not increase the throughput of the overall system.
- There are no special considerations to have when designing a system in order for it to scale vertically.

Solution

You can increase your computing resources by adding memory to your machine or by increasing the number of cores your machine has.

Horizontal Scalling

Context

You have variable usage of your infrastructural resources. You may need to quicly increase or decrease the size of the infrastructure to meet your needs.

Problem

How do you scale your computing capacity in order to respond to the change.

Forces

- If you are using the *Cloud* you have virtually no limit to how many machines you can allocate.
- Horizontally scalable systems are more challenging.
- Having several machines allows you to have some redundancy making your system more robust.
- Having more machines

Solution

You alocate new machines to your project.

Testing

Anti: Manual Testing

Context

You are building a software product and are constantly improving it.

Problem

Whenever you make a change how will you know that the software still works.

Forces

- Changes to even a small functionality may have impact on a completely unrelated functionality of the system.
- Deploying new features without testing the entire system may result in failure.
- Manually testing all aspects of your application may take a long time.

Solution

You charge your developer with the responsability of making sure that the software is functional after he made the changes. The developer usually develops the changes to the software and then procedes to manually test the system.

Automatic Testing

Context

You are building a software product and are constantly improving it.

Problem

Whenever you make a change how will you know that the software still works.

Forces

- Changes to even a small functionality may have impact on a completely unrelated functionality of the system.
- Deploying new features without testing the entire system may result in failure.
- Manually testing all aspects of your application may take a long time.
- Developing automatic tests is additional work and may increase the time needed to develop a feature.

- Some functionalities of your software may not be testable.

Solution

You charge your team with the responsability of developing automated tests(unitary and/or integration and/or functional) for your software. The code is developed and with it the new tests that test that the new changes are working.

Continuous Integration

Continuous Integration

Context

You have several people/teams contributing to the same codebase.

Problem

Whenever a change is made to your project you want to know that the change did not break anything.

Forces

- Whenever something stops working you want to know when did it stop.
- Having quick feedback may help you figure out a solution faster because you still remember what you have done.

Solution

You use an automatic system of *Continuous Integration* that *Build*s, *Test*s the new version of the software and provides *Feedback*.

Build

Context

Your have a change in your software and.

Problem

How do you construct new shipable version of your project ?

Solution

If you have a *Container* and are able to *Download the container* you can obtain the container with the new version of the software . The same can be done using *Virtual Machine* and *Download the Virtual Machine* .

Test

Problem

Whitin your *Continuous Integration* you have successfully created and built the software.

Context

You want to know that the build you generated works.

Solution

Use the *Automatic Testing* and run your entire suit of tests within the new build.

Notify

Problem

You have *Build* and run all the *Test*s in your *Continuous Integration* system.

Context

You want to communicate the result of those steps to your team.

Solution

You can communicate the result of the integration by using a *Chat Tool* or by sending an *Email* to the developer/team.

Version Control

Git Flow

Context

You are using Git or a similar version control software for your code.

Problem

You want to organize your version control flow in a way that you separate what is being develop from what has been developed and from the current release/production code.

Solution

You can choose to divide your project in 3 branches. The master branch will represent a release or the version that is currently in production. The develop branch will represent all of the current developed features. For each new functionality a new branch will be created. Once terminated the feature if accepted (through pull request) will be merged into the develop branch. When you want to signal a new release you just merge the desired state of the develop branch into the master branch.

Feature Branches

Context

You are using Git or a similar version control software for your code.

Problem

You want to organize your version control flow so that you separate what has been done from what is still being done.

Solution

You have a master branch. For each new feature you create a new branch. Once the feature is done you merge it into the master branch.

Teams

Anti: Specialized Teams

Context

You have a set of professionals that you want to allocate to one or several projects.

Problem

You want to organize them in a productive way.

Solution

You divide them by specializations and have them work separately.

Multidisciplinary

Context

You have a set of professionals that you want to allocate to one or several projects.

Problem

You want to organize them in a efficient way that allows them to quicly adapt to new problems.

Solution

You create teams that include individuals from different disciplines.

Keep it Small

Context

Problem

You do not know how many people to allocate to the project team.

Solution

Depending on your needs you could allocate between 1 and 9 people.

Keep it loose

Context

Problem

You do not know how to hierarchical organize people to the project.

Solution

You create an horizontal structure inside your team and let them organize themselves.

Communication

The direct communicator

Context

People with different backgrounds and degrees of knowledge are working together.

Problem

You want knowledge and ideas to be shared between people.

Solution

Allow you team members to communicate directly.

Chat is for links files and winks

Context

Several people are working in the same project.

Problem

People want to share links and files.

Solution

Most chat tools allow for the sharing of files and links between members.

Jobs

Anti: Daemon

(Confirmar com EZ4U)

Context

Your application has some maintenance or some backgrounds tasks that take a significant ammount of time to complete.

Problem

How do you do it?

Solution

You create a Daemon that sequencially performs each tasks and notifies some service or updates a service when the task is completed.

Launch new infrastructure

Problem

Your application has some maintenance or some backgrounds tasks that take a significant ammount of time to complete.

Solution

You sequencially launch a new piece of infrastructure to perform the task you have to perform.

Monitoring

Monitor the present

Context

You have several servers.

Problem

You want to know wich ones are on and which ones are not.

Solution

You deploy a tool in each infrastrucuture element that tells a master one if that server is online or not.

Save the past

Context

You have several servers.

Problem

You want to know when there is a problem, what has happened.

Solution

You deploy a tool in each infrastrucuture element that reports to the master one the logs from your applications.

Alert

Thresholding

Context

You have a need to watch over your infrastructure and you want to fix problems when they appear.

Problem

You do not want to have to have someone continuously looking to find possible problems.

Solution

Define indicators that try to assess your infrastructure . Define thresholds and alert levels for each indicator. When a treshold is exceeded an alert should be triggered.

Anti: Wake the devs

Context

You have defined tresholds and alert levels for your infrastructure.

Problem

You don't know who to contact.

Solution

You send an alert to everyone involved in the project.

Rotate and Wake

Context

You have defined tresholds and alert levels for your infrastructure.

Problem

You don't know who to contact.

Solution

You have an on call engineer and you rotate the position through the team. The notification is sent to that engineer.

Error Handling

Rolling Back

Context

You have detected a problem in your production application.

Problem

You want to solve that problem.

Solution

You change the production version of your application to previously working version.

Rolling Back: DNS Switching

Context

You have detected a problem in your production application.

Problem

How do you change the live version to the previous one ?.

Solution

You change the DNS for it to point to the old version of the application.

Rolling Back: Deploy again

Context

You have detected a problem in your production application.

Problem

You want to solve that problem with minimu.

Solution

You do a normal deploy but choose the previous version.

Hotfix

Context

You have detected a problem in your production application.

Problem

You want to solve that problem with minimu.

Solution

You make the changes necessary to correct your application and then deploy this new version.

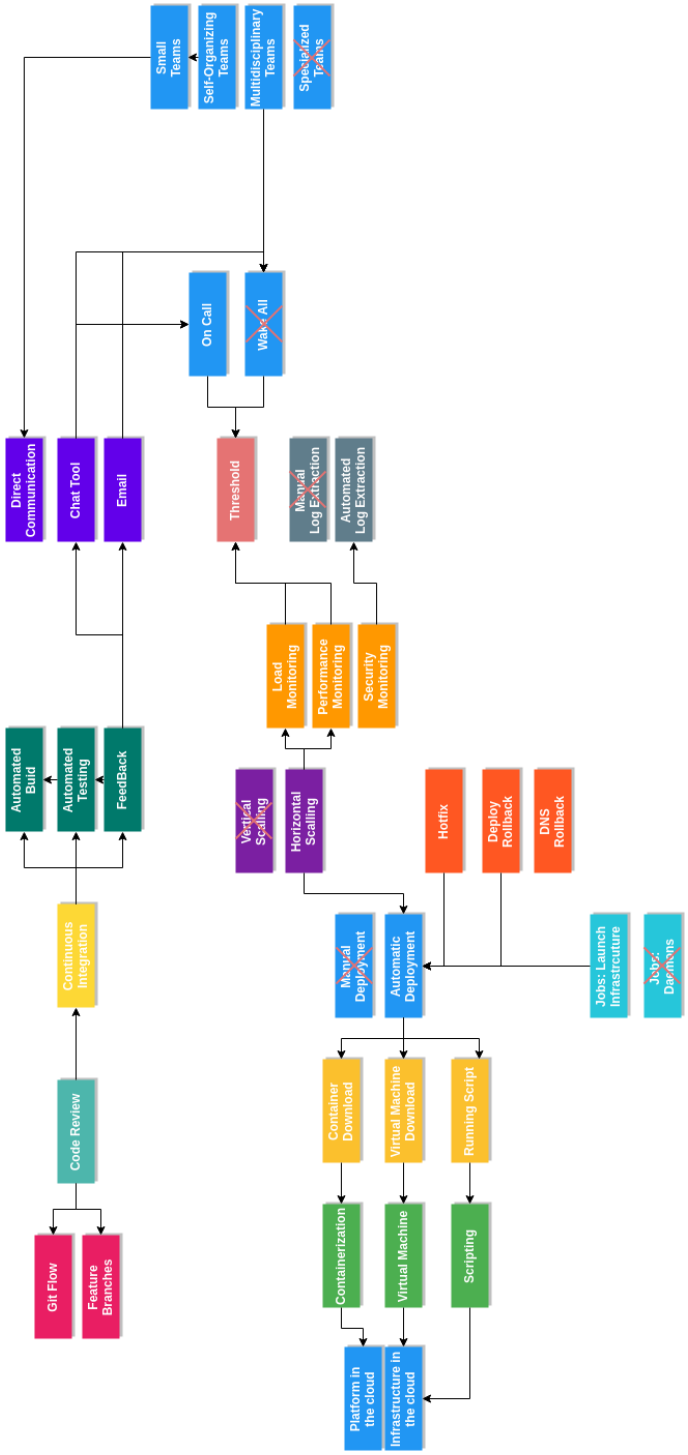

Figure 1.1: Pattern Map