



# Trunk Based Development

Kamil Mówiński - STXNext

# Agenda

The background features abstract geometric shapes. A large teal 'X' shape is formed by two overlapping diagonal bars. To the right of the teal bars, there are two grey triangular shapes pointing downwards, one in the upper right and one in the lower right.

# Agenda

What?

# Agenda

What?

When?

# Agenda

What? When? Where?

# Agenda

What?

When?

Where?

Why?

# Agenda

What?

When?

Where?

Why?

Who?

# Agenda

What? When? Where? Why? Who?



# Agenda

What? When? Where? Why? Who?

# Agenda

The background features a large, stylized 'X' shape. The left half of the 'X' is a teal color, and the right half is a light grey color. The 'X' is formed by two thick diagonal bars that meet in the center.

What?

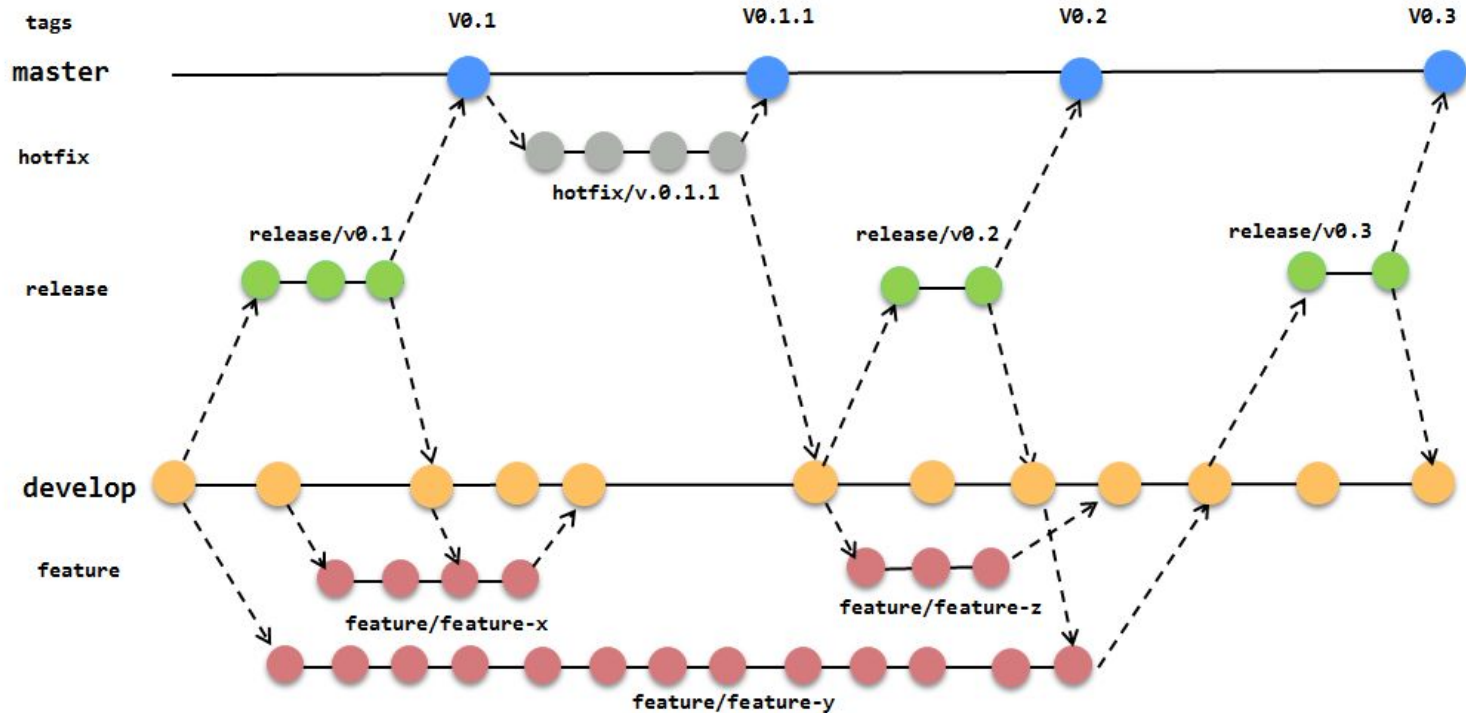
Why?

How?

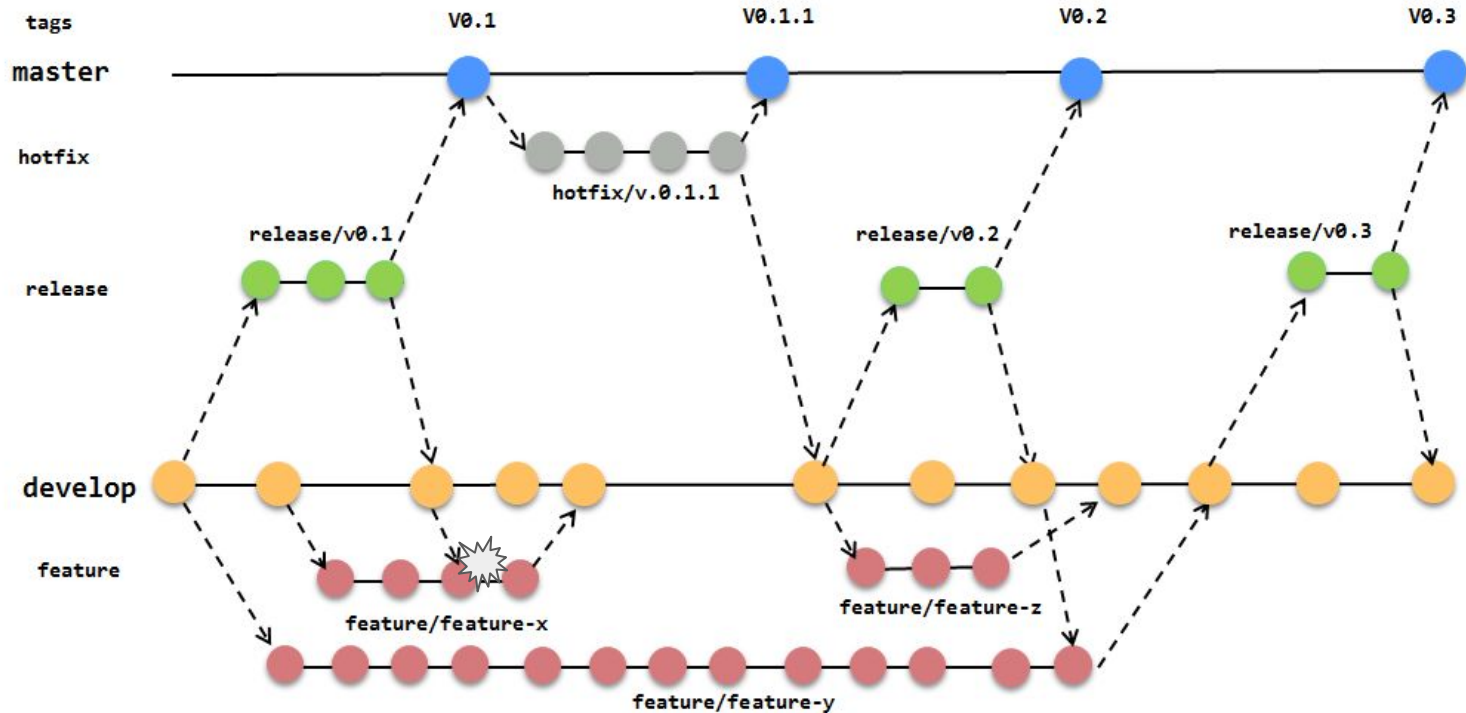
# Agenda

1. What “Trunk Based Development” is?
2. Why should I persuade my company/team to work with TBD?
3. How can I work with daily pushing?
4. Q&A session

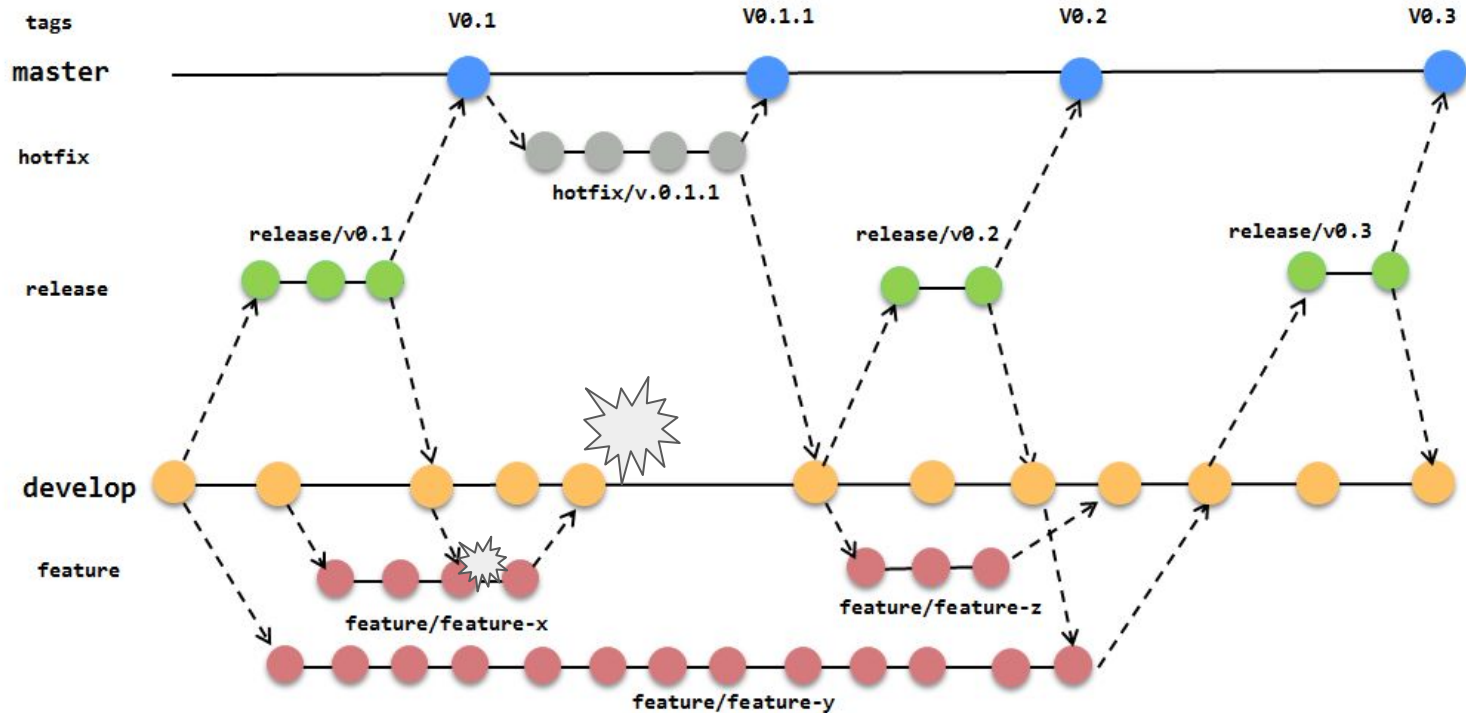
# Feature Based Development - What is it?



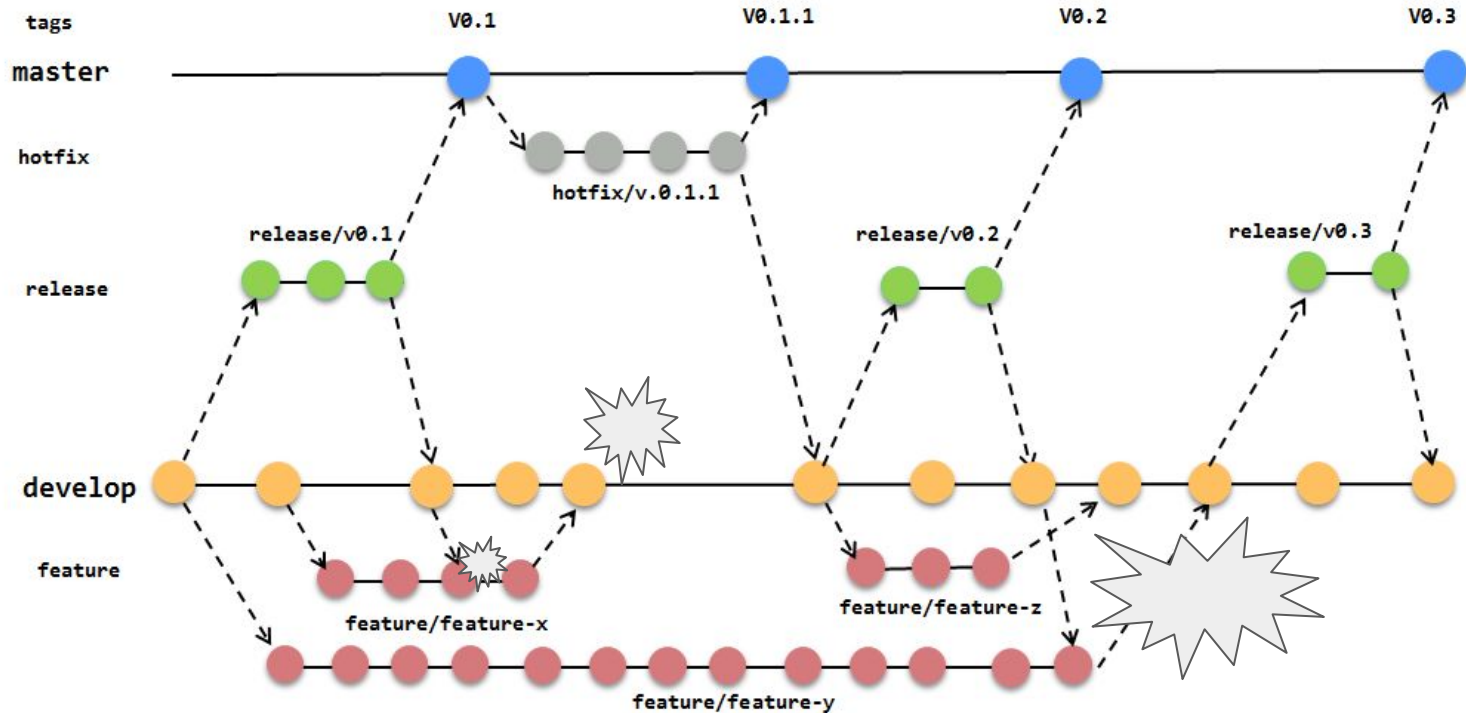
# Feature Based Development - What is it?



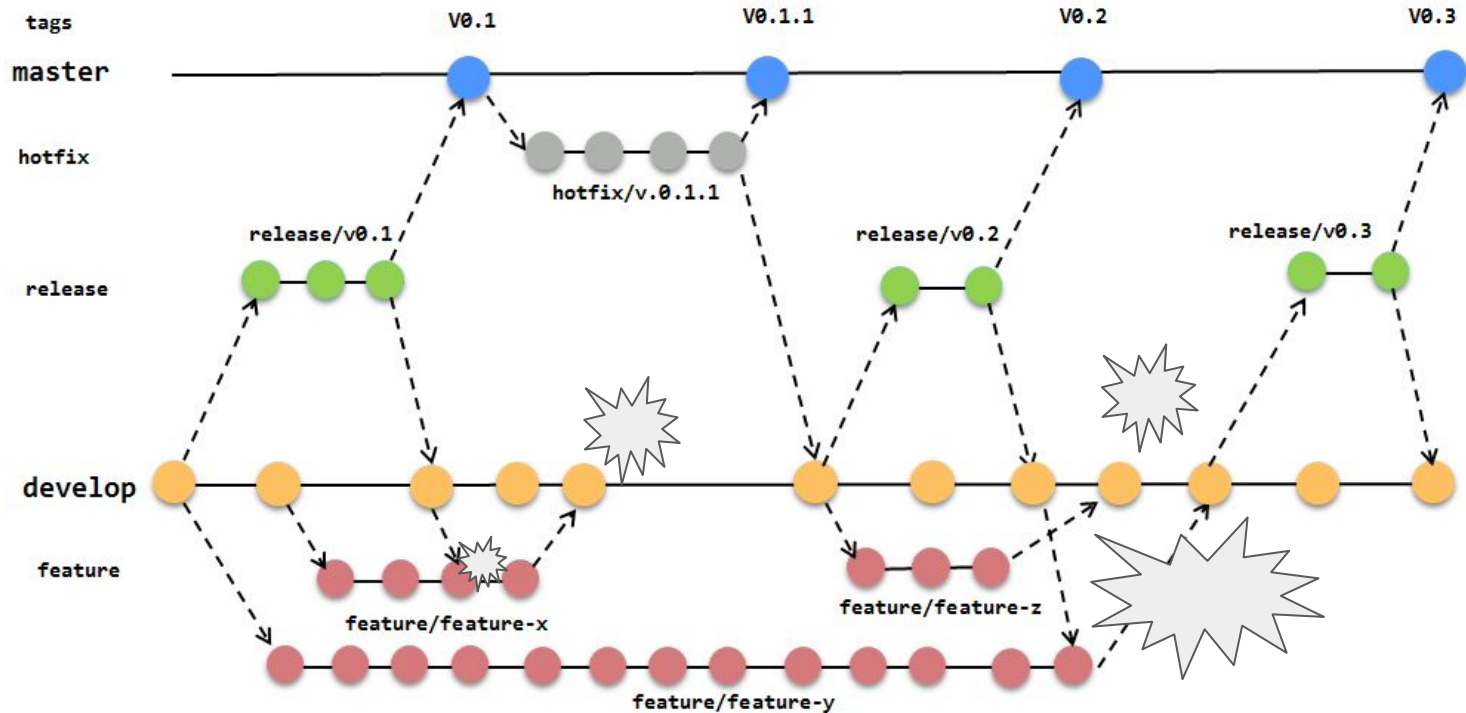
# Feature Based Development - What is it?



# Feature Based Development - What is it?

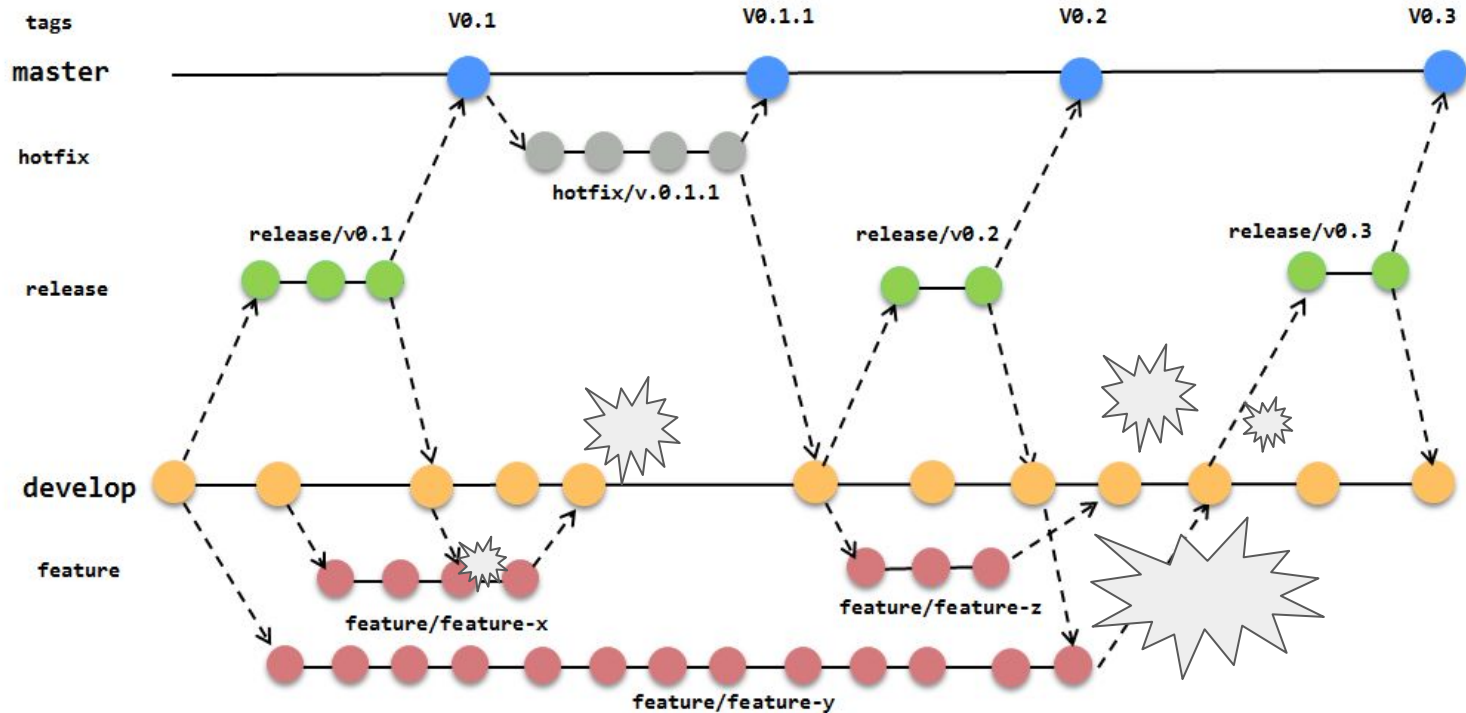


# Feature Based Development - What is it?





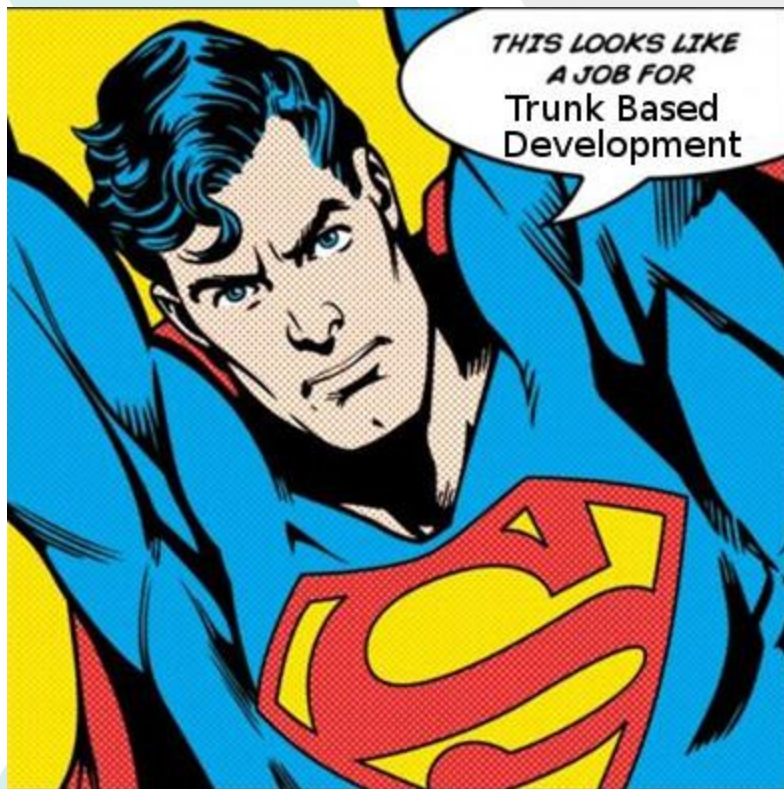
# Feature Based Development - What is it?



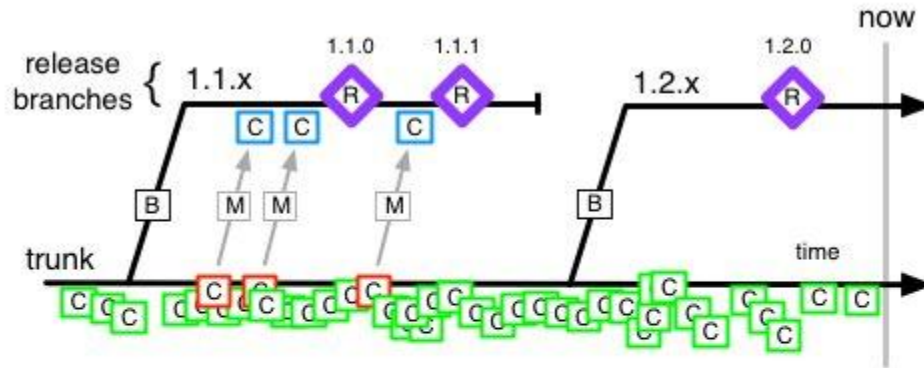
```
[22:02][system][nick:~/hacking/newast_mirah(old_newast)]$ git merge master
error: refusing to lose untracked file at 'lib/mirah/typer/simple.rb'
error: refusing to lose untracked file at 'javalib/mirah-bootstrap.jar'
Auto-merging test/test_helper.rb
CONFLICT (add/add): Merge conflict in test/test_helper.rb
Auto-merging test/plugins/gwt_test.rb
CONFLICT (content): Merge conflict in test/plugins/gwt_test.rb
Auto-merging test/jvm/rescue_test.rb
CONFLICT (add/add): Merge conflict in test/jvm/rescue_test.rb
Auto-merging test/jvm/main_method_test.rb
CONFLICT (add/add): Merge conflict in test/jvm/main_method_test.rb
Auto-merging test/jvm/macros_test.rb
CONFLICT (add/add): Merge conflict in test/jvm/macros_test.rb
Auto-merging test/jvm/jvm_compiler_test.rb
CONFLICT (content): Merge conflict in test/jvm/jvm_compiler_test.rb
Auto-merging test/jvm/javac_test_helper.rb
CONFLICT (content): Merge conflict in test/jvm/javac_test helper.rb
```







# Trunk Based Development



# Core rules of Trunk Based Development





# Core rules of Trunk Based Development





# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day

**EVERYDAY**  
 **I'M**  
**COMMITIN'**

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review
- Local branches should be as short-living as possible

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review
- Local branches should be as short-living as possible
- HotFix also commits into mainline branch

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review
- Local branches should be as short-living as possible
- HotFix also commits into mainline branch
- Cherry-picked hotfix to Release branch

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review
- Local branches should be as short-living as possible
- HotFix also commits into mainline branch
- Cherry-picked hotfix to Release branch
- Only Release Manager can branch release branches

# Core rules of Trunk Based Development

- Everyone commits to mainline branch (trunk, master, develop)
- Everyone commits every day
- Developers can use local branches e.g. to review
- Local branches should be as short-living as possible
- HotFix also commits into mainline branch
- Cherry-picked hotfix to Release branch
- Only Release Manager can branch release branches
- Developers do not break the build with any commit.

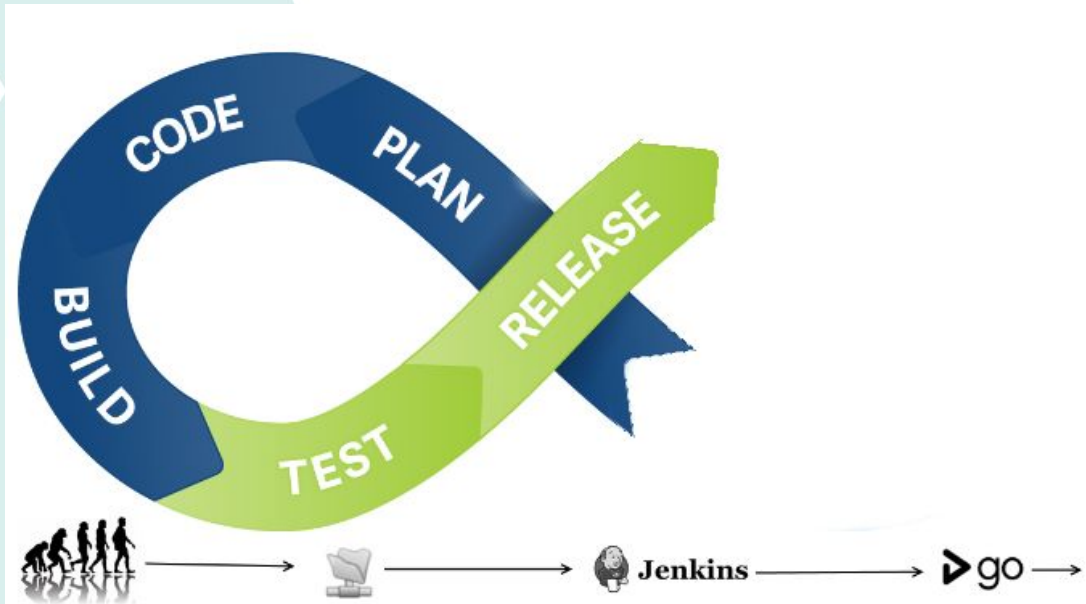


The background features abstract geometric shapes. A large teal chevron points downwards from the top left. To its right, a grey chevron points upwards from the bottom right. Another grey chevron points downwards from the top right. The text is centered in the upper portion of the image.

Why should we use Trunk Based Development?

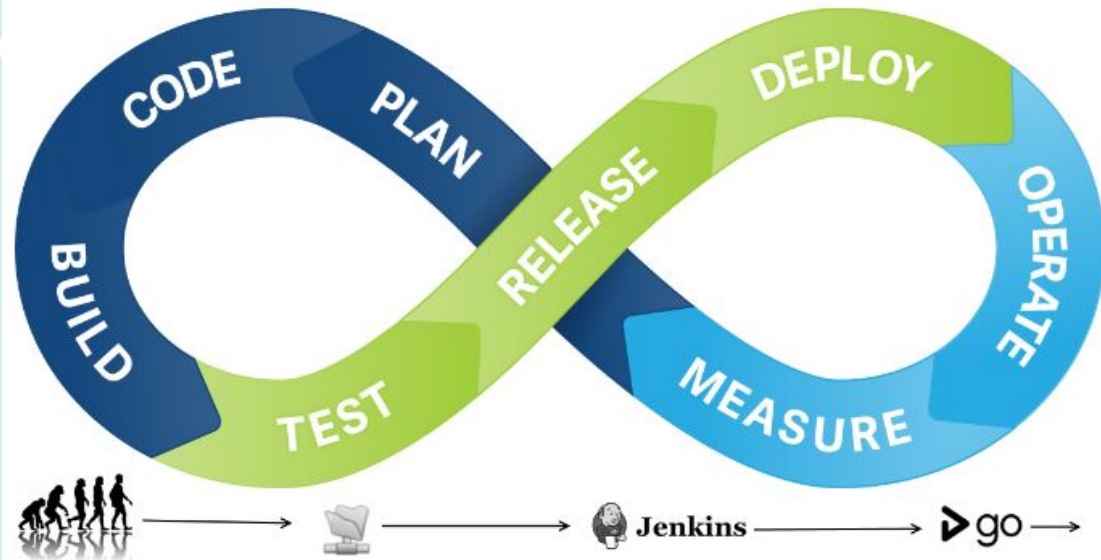
# Why should we use Trunk Based Development?

- Continuous integration



# Why should we use Trunk Based Development?

- Continuous integration
- Continuous delivery



# Why should we use Trunk Based Development?

- Continuous integration
- Continuous delivery
- No merge hell



# Why should we use Trunk Based Development?

- Continuous integration
- Continuous delivery
- No merge hell
- Release branches - piece of cake



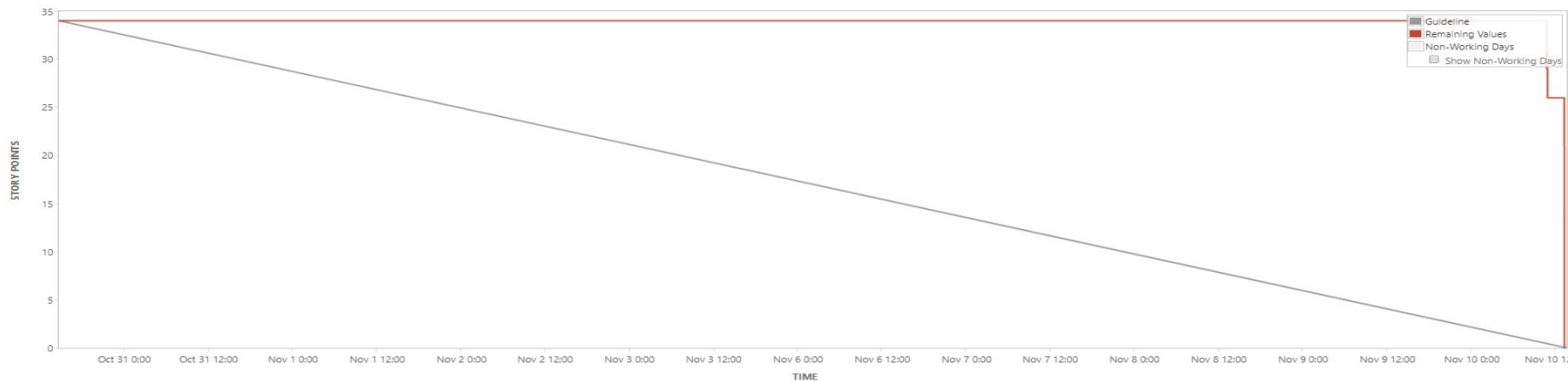
# Why should we use Trunk Based Development?

- Continuous integration
- Continuous delivery
- No merge hell
- Release branches - piece of cake
- Release when you want



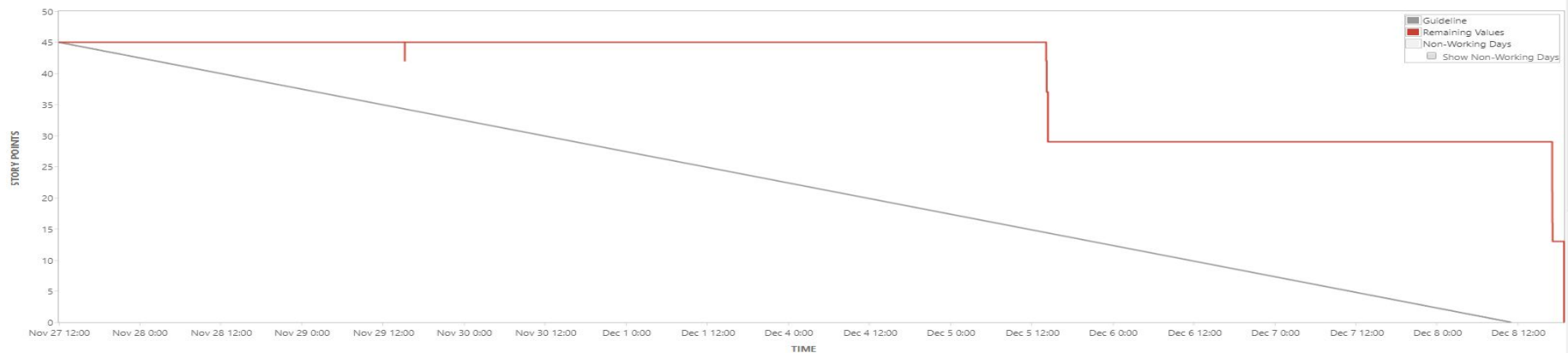
Roll-out the first version of the Time Widget for Projects!

31.10.2017 - 10.11.2017



27.11.2017 - 08.12.2017

Teams in new app - create and view features delivered



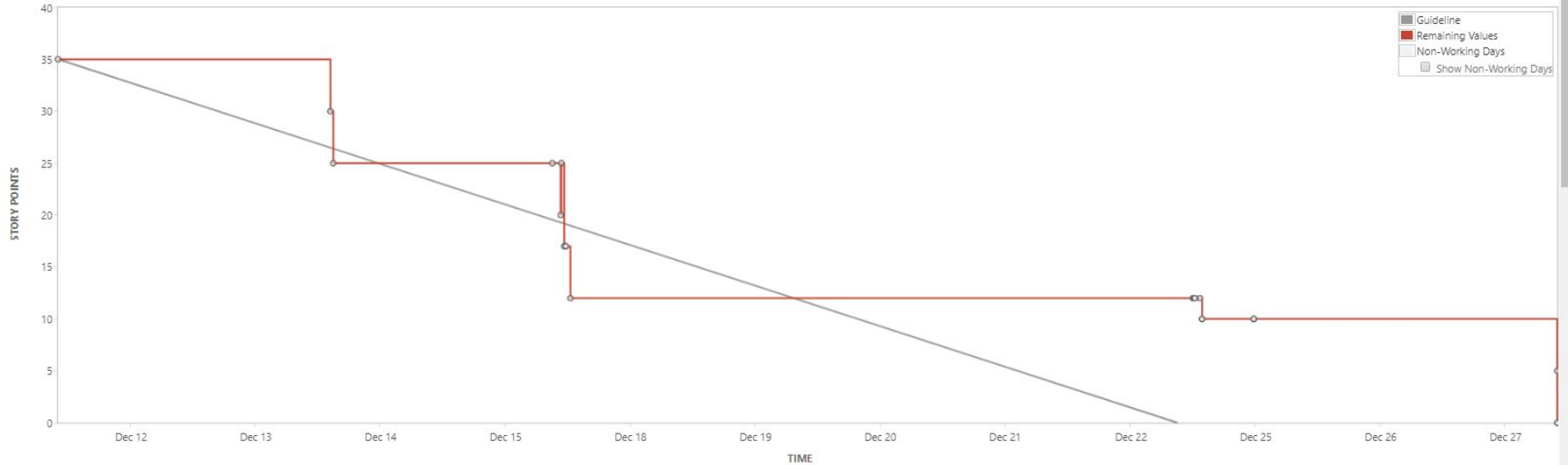
# Why should we use Trunk Based Development?

- Continuous integration
- Continuous delivery
- No merge hell
- Release branches - piece of cake
- Release when you want
- Look at my burndown charts - my charts are amazing!



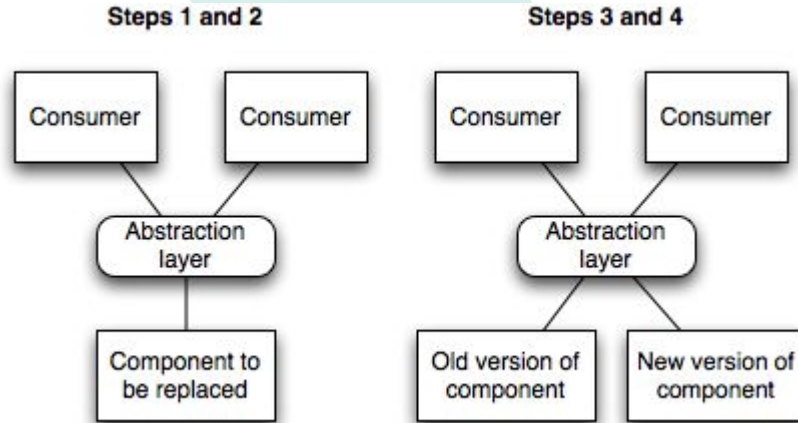
12.12.2017 - 27.12.2017

Deliver Teams v1 related features till Dec, 15th



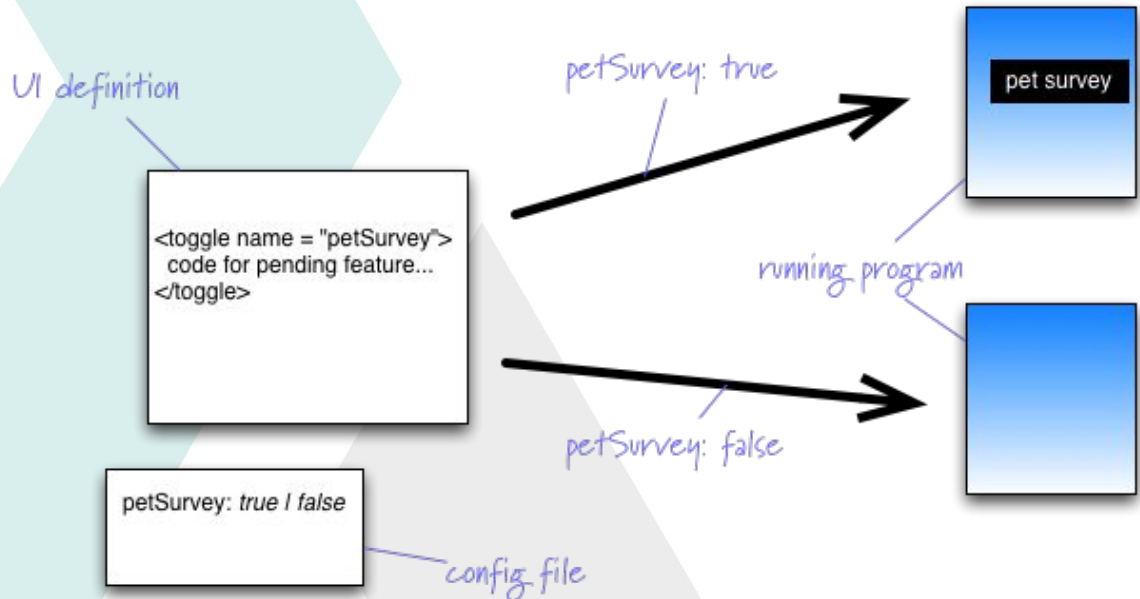
# How to handle it?

- Branch by Abstraction



# How to handle it?

- Branch by Abstraction
- Feature Toggles



# Database migration

The background features abstract geometric shapes. A large teal chevron points from the top-left towards the center. To its right, a grey chevron points from the top-right towards the center. Below these, two more grey chevrons point from the right side towards the center, creating a sense of depth and movement.

# Database migration

- Changes must perform backward compatibility

# Database migration

- Changes must perform backward compatibility
- Only one SQL statement in migration

# Database migration

- Changes must perform backward compatibility
- Only one SQL statement in migration
- Should be integrated into build process

# Useful library for Python

**Django-Waffle**

<https://waffle.readthedocs.io>

**Django-Feature-Flipper**

<https://github.com/mypebble/django-feature-flipper>

**Flask-FeatureFlags**

<https://github.com/trustrachel/Flask-FeatureFlags>

**Flagon**

<http://www.oss.io/p/ashcrow/flagon>



# Useful library for JavaScript

**Javascript Feature Flags**

<https://github.com/jayf/javascript-feature-flags>

**Angular Feature Flags**

<https://github.com/michaeltaranto/angular-feature-flags>

**Ember-Feature-Flags**

<https://github.com/kategengler/ember-feature-flags>

**Node Feature Flags**

<https://www.npmjs.com/package/feature-toggles>

# Bibliography

- [Trunk Based Development](#)
- [What is Trunk-Based Development?](#)
- [Enabling Trunk Based Development with Deployment Pipelines](#)
- [Branch By Abstraction](#)
- [Branch By Abstraction by Martin Fowler](#)
- [Feature Flags](#)

The background features abstract geometric shapes. A large teal 'X' shape is centered, with two grey triangles pointing towards the center from the top and bottom right.

# Questions and Answers

```
→ thanks git:(master) git merge origin/master
Auto-merging main.cpp
CONFLICT (zawartość): Merge conflict in main.cpp
Automatic merge failed; fix conflicts and then commit the result.
→ thanks git:(master) X git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
```

```
Unmerged paths:
  (use "git add <file>..." to mark resolution)
```

```
      both modified:   main.cpp
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
→ thanks git:(master) X cat main.cpp
#include <iostream>
```

```
int main(int argc, char *argv[]) {
<<<<<< HEAD
    std::cout << "Thank you" << std::endl;
=====
    std::cout << "Thanks" << std::endl;
>>>>>> origin/master
    return 0;
}
```

```
→ thanks git:(master) X █
```