# Don’t do what I did

Yossi Moalem

© Fortinet Inc. All Rights Reserved.

## Case 1

© Fortinet Inc. All Rights Reserved.

## The scenario

- Low level visualization function
- In edge cases, displayed the object rotated

- Fix the bug

© Fortinet Inc. All Rights Reserved.

© Fortinet Inc. All Rights Reserved.

## The problem

- Not as rare as I would liked it to be
  - Other parts of the application accommodated for this bug

- Revert it

© Fortinet Inc. All Rights Reserved.

# Sync with everyone and fix at the same time

- Can work on small scale
- Can work if one person can do the work

- Different teams – different priorities

- If one team fails – we should revert the whole fix

© Fortinet Inc. All Rights Reserved.

## Gradual migration

```
void drawObject ( … ) {
    //faulty Implementtion;
}
```

```
void drawObjectEx ( … ) {
    // correct Implementation;
}
```

## Two  different implementations

```
void drawObject ( … ) {
      //faulty Implementtion;
}
```

```
void drawObjectEx ( … ) {
      // correct Implementation;
}
```

## Call the first implementation:

```
void drawObject ( … ) {
    //faulty Implementtion;
}
```

```
void drawObjectEx ( … ) {
     drawObject( … );
    //accommodate for the bug
}
```

## Call the new one

```
void drawObject ( … ) {
     drawObjectEx( …. );
       //Recreate the bug
}
```

```
void drawObjectEx ( … ) {
     // correct Implementation;
}
```

© Fortinet Inc. All Rights Reserved.

## Names

```
void drawObject ( ... ) {
    drawObjectEx( …. );
      //Recreate the bug
}
```

```
void drawObjectEx ( ... ) {
    // correct Implementation;
}
```

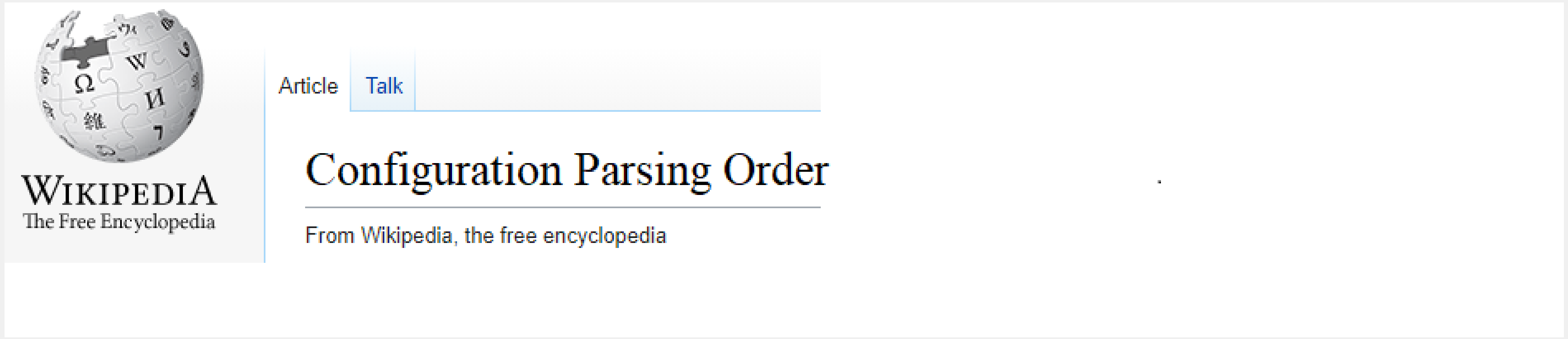
## Not only bug fixes - Hyrum's Law

With a sufficient number of users of an API,
it does not matter what you promise in the contract:
all observable behaviors of your system
will be depended on by somebody.

- Order the configuration was parsed was changed
- Old order was dependent upon

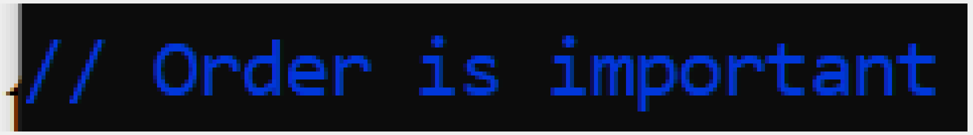
© Fortinet Inc. All Rights Reserved.

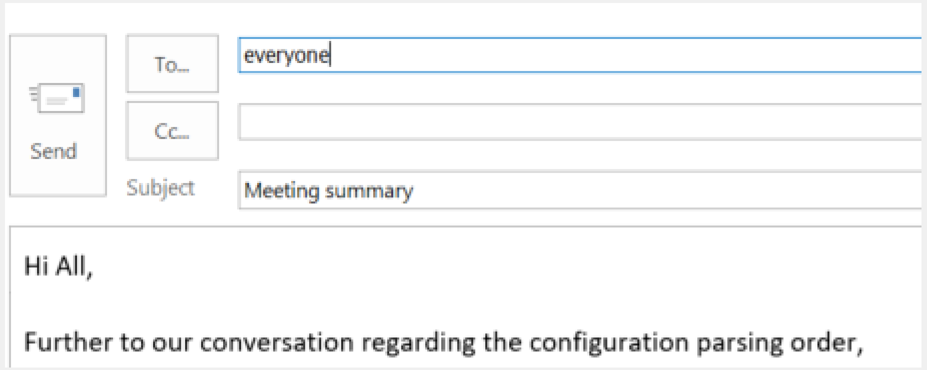
# Document the assumptions


Article Talk

Configuration Parsing Order

From Wikipedia, the free encyclopedia


```
// Order is important
```



To: everyone

Subject: Meeting summary

Hi All,

Further to our conversation regarding the configuration parsing order,

## Make the assumptions explicit

```
sort (begin(rules), end(rules),

            [ ] ( const Rule & lhs, const Rule & lhs )
            { return lhs.priority < rhs.priority; } );
```

© Fortinet Inc. All Rights Reserved.

## Takeaways

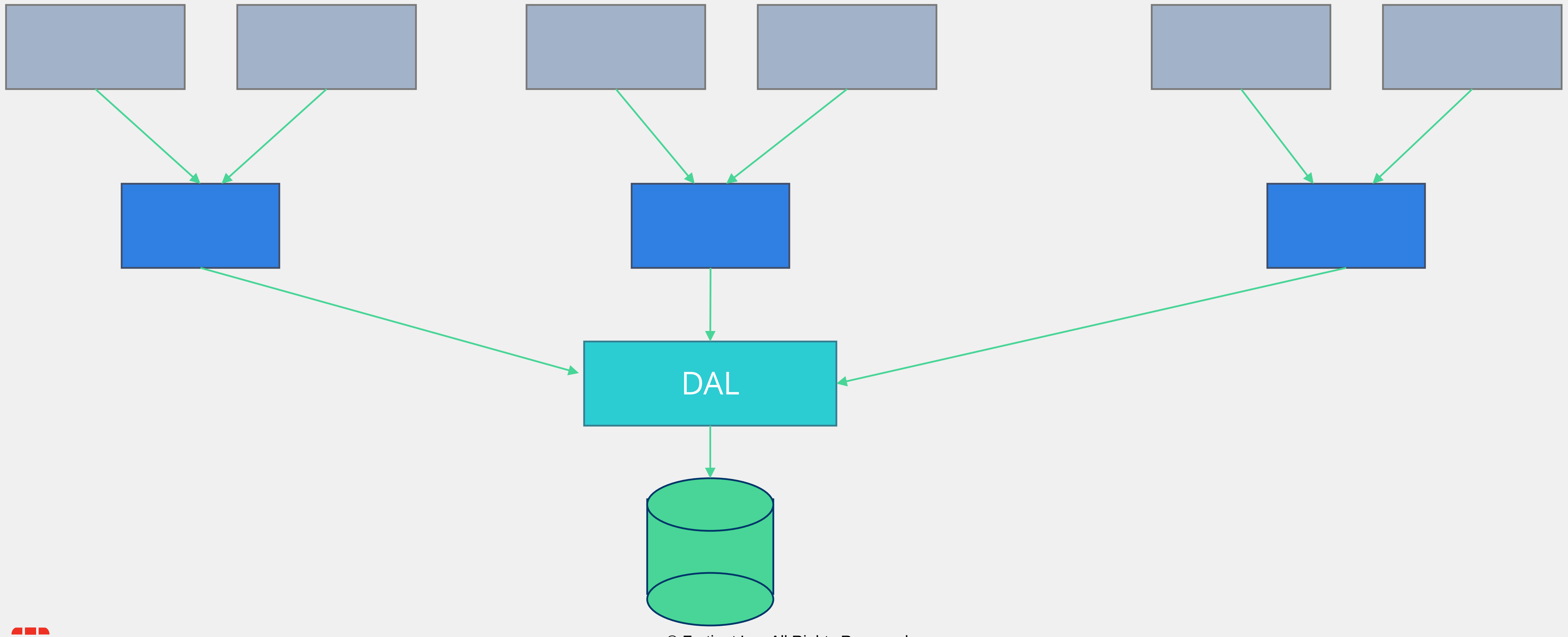
- Your fix will affect other parts of the application
  - The lower the fix – the more risky it is
- Prepare migration path
  - End with good, clean code
- Document any assumption
  - In code

© Fortinet Inc. All Rights Reserved.

## Case 2

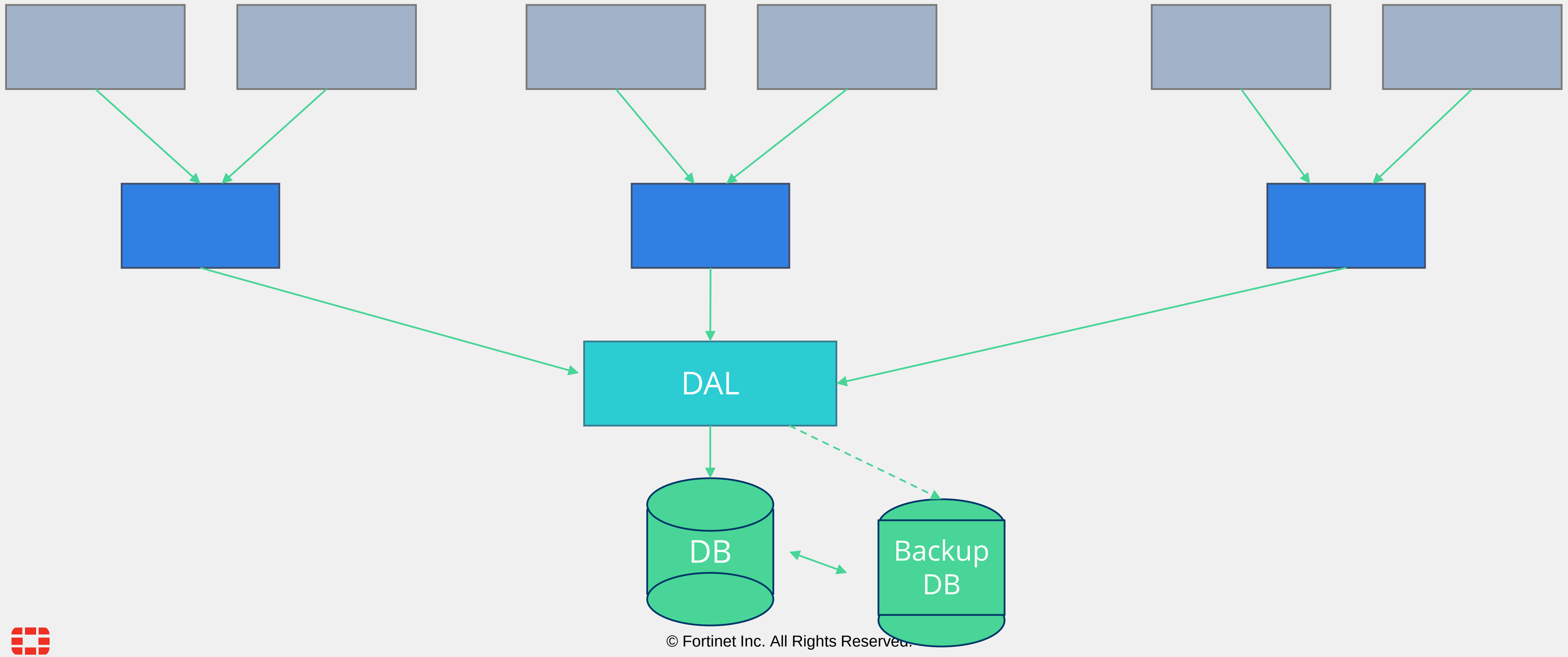
© Fortinet Inc. All Rights Reserved.

## The architecture



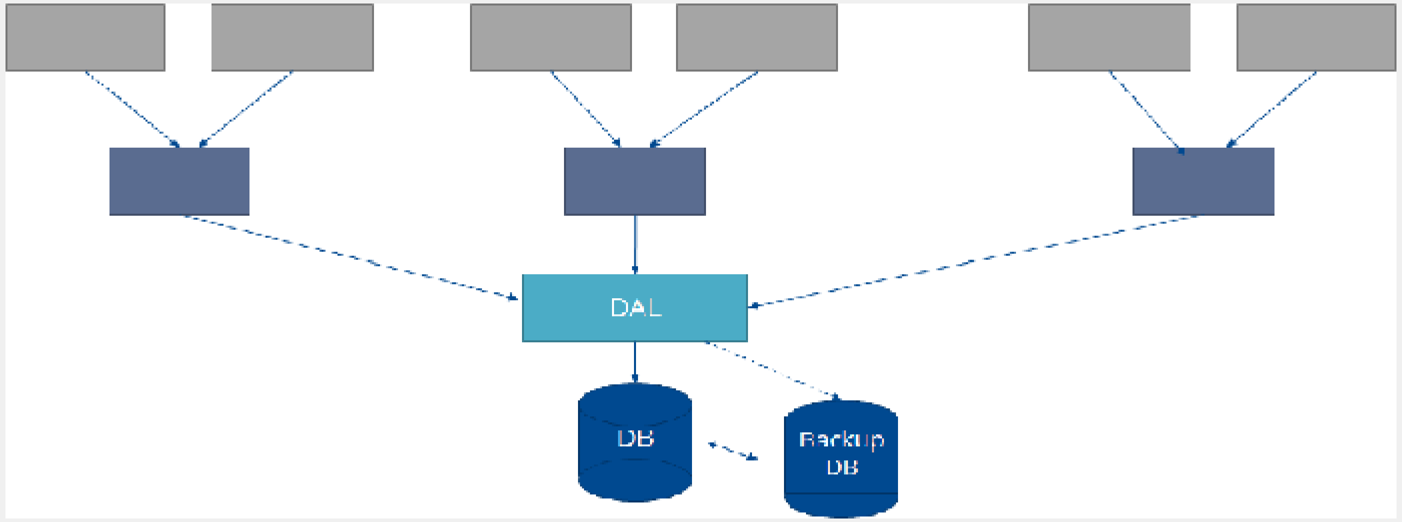
## The Requirement

Process crash on DB roll back attempt



## Possible solutions:

- Move the connection dependent code to DAL
  - Requires process familiarity
- Add observer
  - Very technical solution



© Fortinet Inc. All Rights Reserved.

## Resolving with observer:

- Many more edge
- Needed another observer.
- Well, actually 2 of them

- And it was over
  - It wasn’t

This was obviously the wrong choice!

© Fortinet Inc. All Rights Reserved.

## When to stop:

- Set a limit
  - Time, amount of bugs.
- When this time ends,
  - If possible stop altogether.
  - Or, go back to the scratching board
- And stand by it
  - This is easier said than done….

© Fortinet Inc. All Rights Reserved.

## Takeaways:

- Not knowing the code is not a reason for sub-optimal decisions
  - Never decide unless you DO know the code
- If you keep finding errors in the solution – reconsider it
- Throwing away bad solution and restarting is not a failure

© Fortinet Inc. All Rights Reserved.

## The re-write:

- Too risky!

## Case 2

© Fortinet Inc. All Rights Reserved.

## The scenario

- Component with sub-optimal code
  - Many bugs
  - Hard to debug
- Bug in one of the flows

Refactor the code!

© Fortinet Inc. All Rights Reserved.

## The problem

- Each fix called for another fix
- Ended out with very large patch

- Unsafe to introduce

© Fortinet Inc. All Rights Reserved.

## Limiting the refactoring size

- Repeat forever:
    - Find small changes and fix them

Problem:

May spend a lot of time on unimportant fixes

© Fortinet Inc. All Rights Reserved.

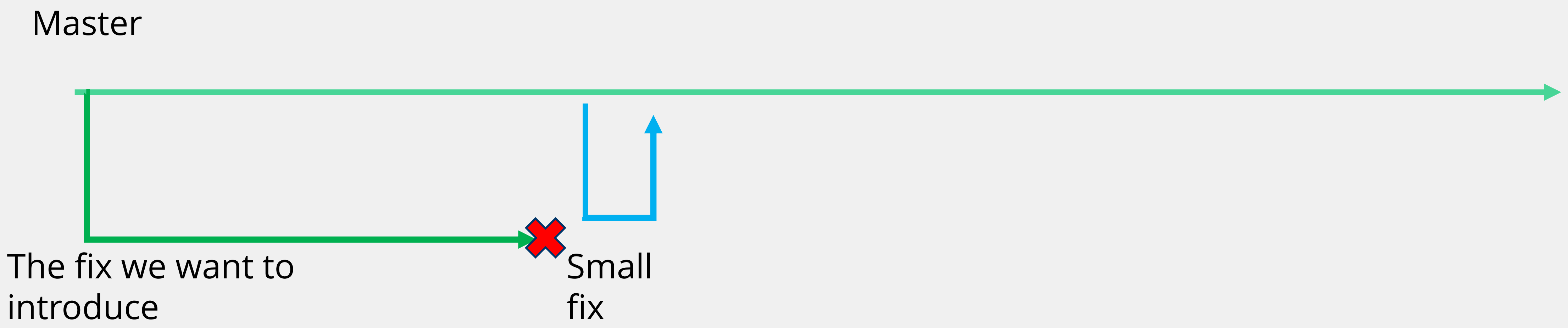
## What we really want:

- Introduce many, small changes
- But limit ourselves to the “important” changes

- “peek” at the issues we would encounter
- And fix only those

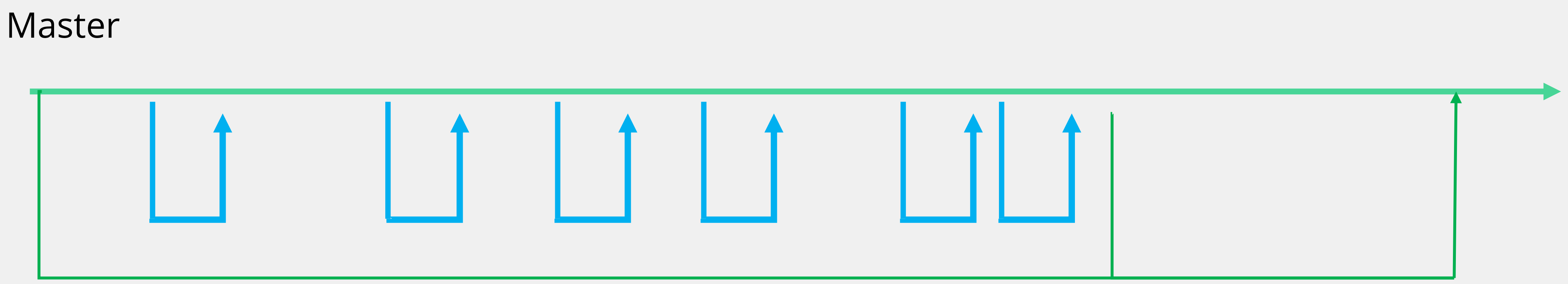
© Fortinet Inc. All Rights Reserved.

# The incremental way



© Fortinet Inc. All Rights Reserved.

## The incremental way



## Takeaways:

- Limit the size of the fix
- Plan for the big fix
  - And push fixes that will help you get there.
- Incomplete fix are fine. As long as they are in the right direction

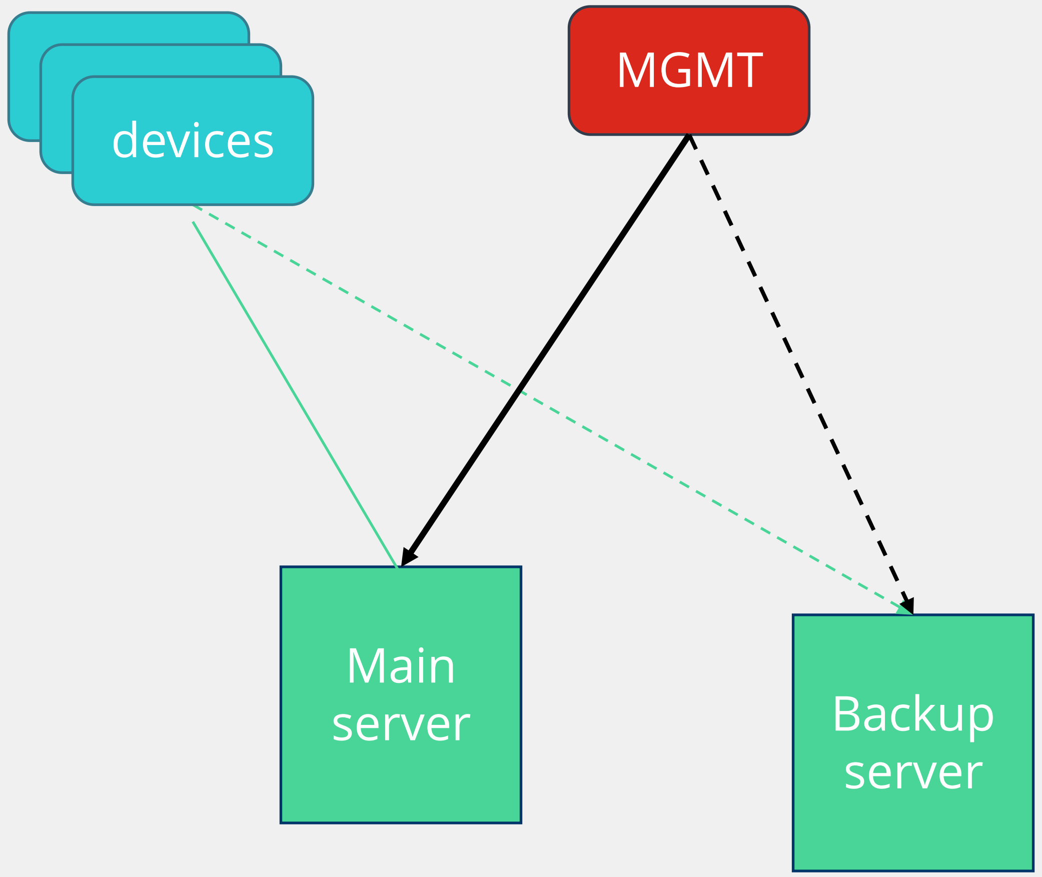
## Design based on technology

rather than need/user

© Fortinet Inc. All Rights Reserved.

## Current architecture

- Server only transfers messages between devices
- Max 20k devices



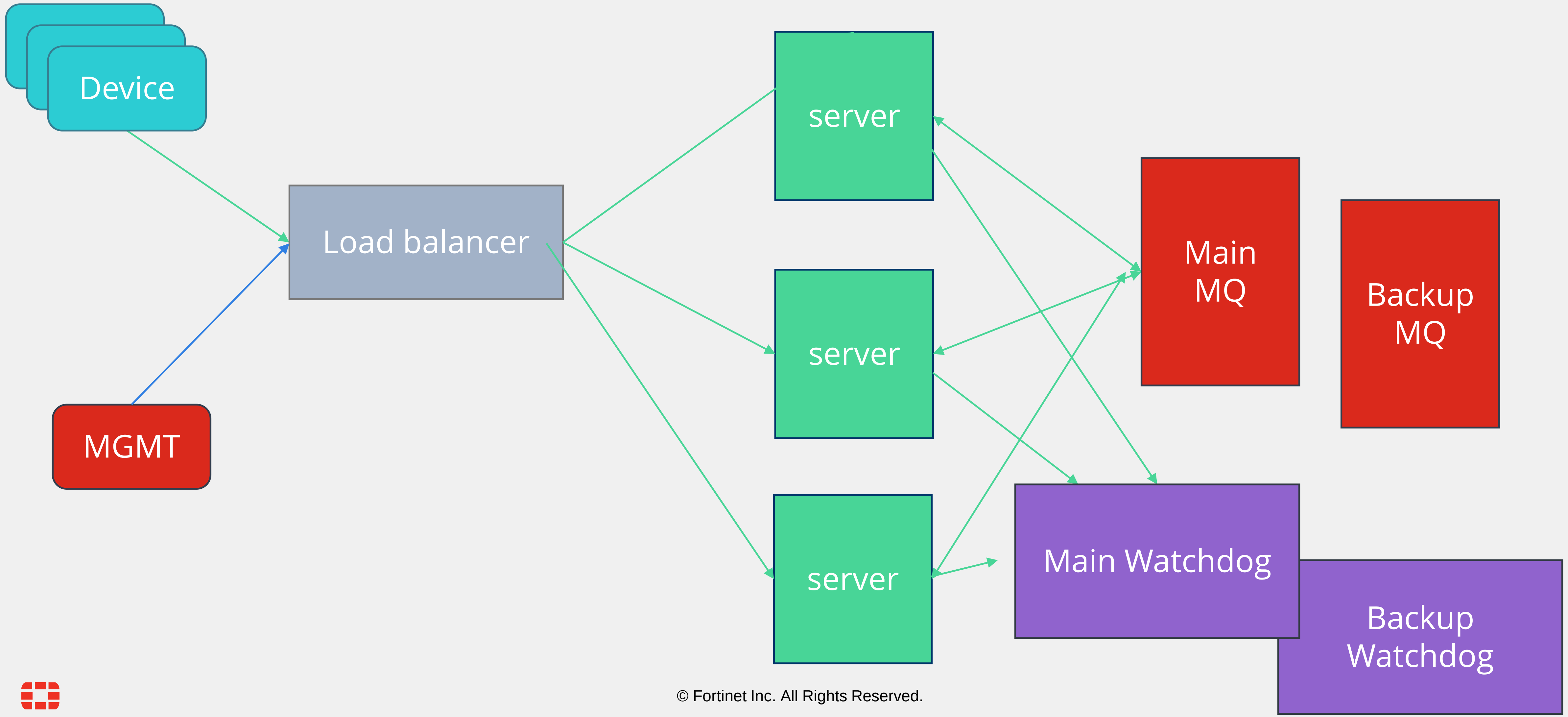
© Fortinet Inc. All Rights Reserved.

## Requirement:

- Unlimited scalability
- Transparent migration

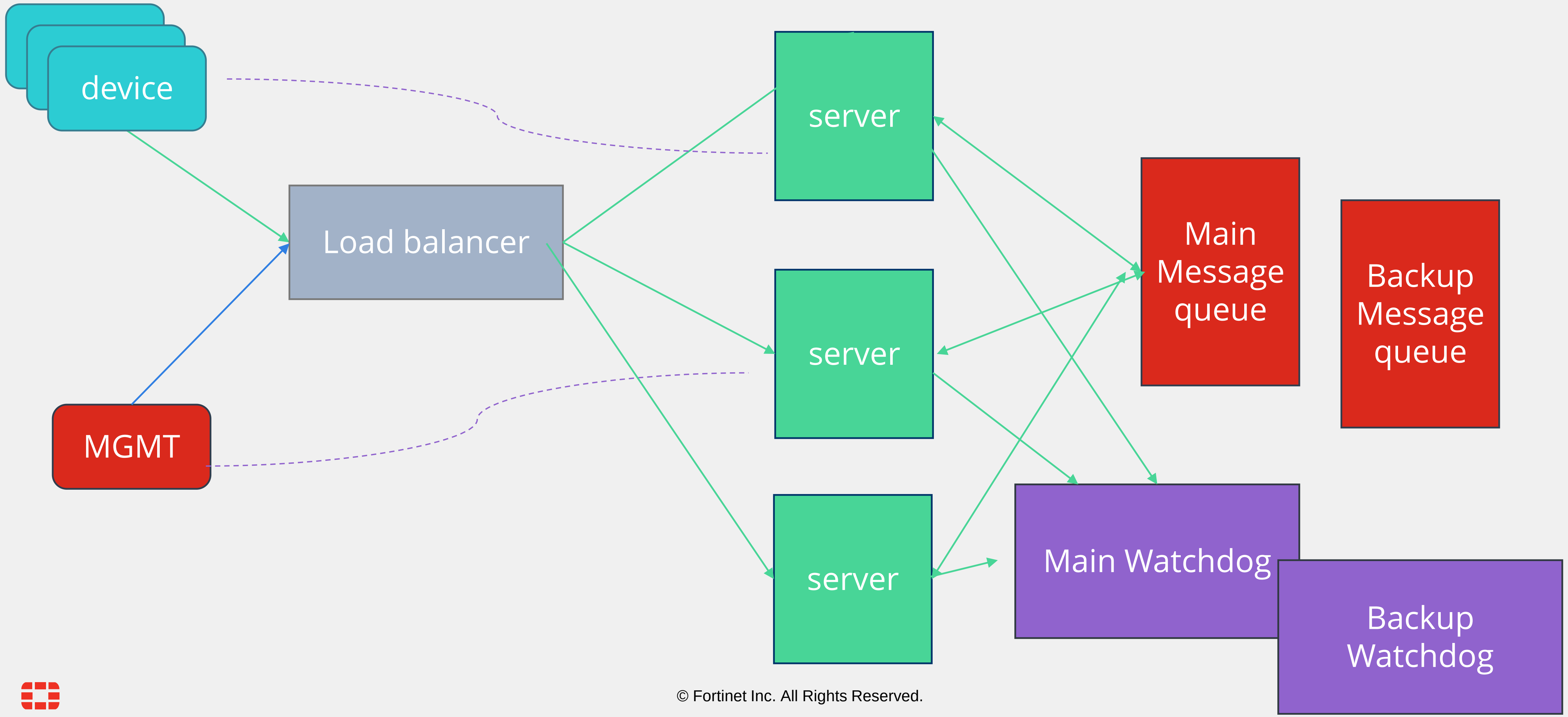
© Fortinet Inc. All Rights Reserved.

# New architecture



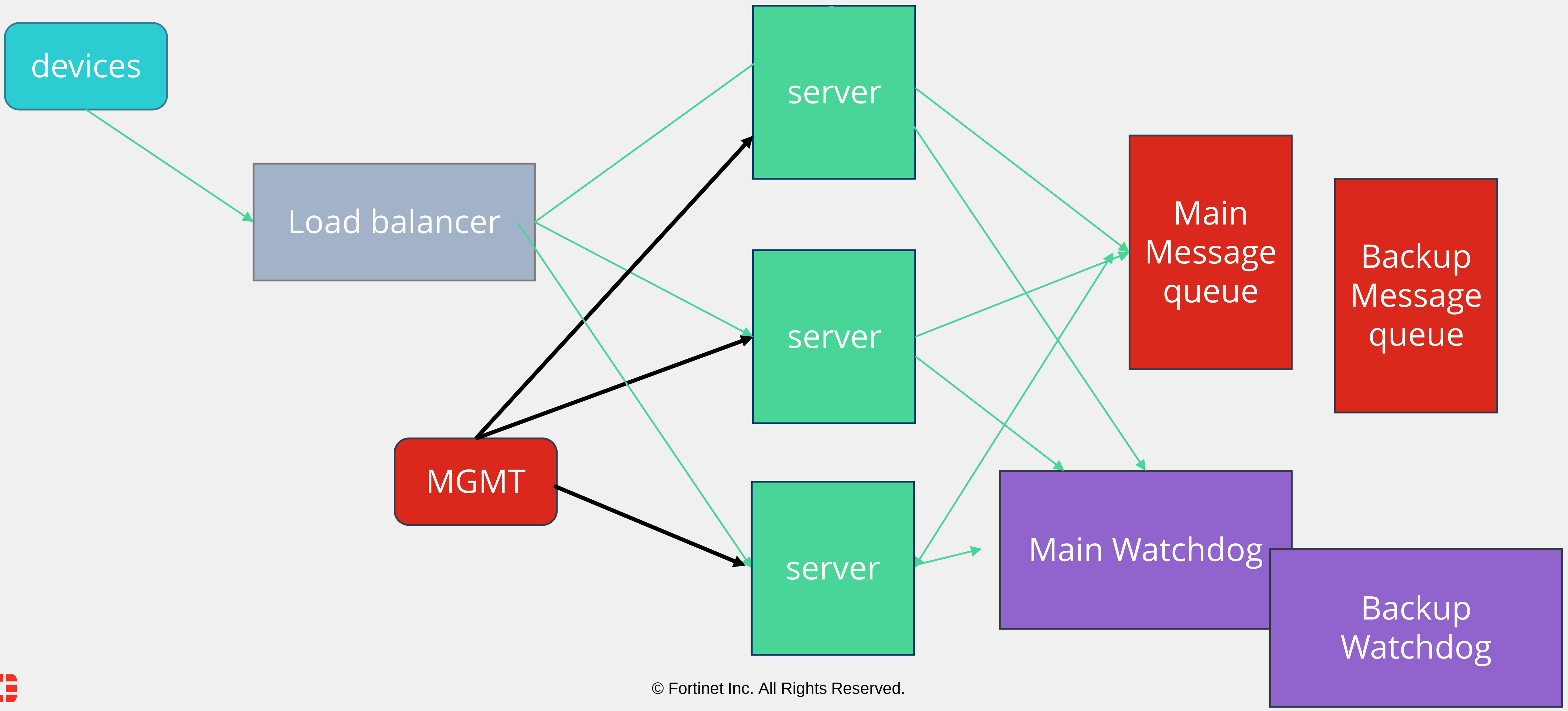
© Fortinet Inc. All Rights Reserved.

# New architecture



© Fortinet Inc. All Rights Reserved.

# New architecture



© Fortinet Inc. All Rights Reserved.

## The twist

- A new, urgent requirement came
- Refactor time!

## Optimization keys:

- Overcommitting
- Load Sharing
- Simplicity
  - Simple containers
  - Simple algorithms
- Optimize for the rare case
- Locking contention
- Copy instead of move/share
- Memory layout

© Fortinet Inc. All Rights Reserved.

## Takeaways:

- Rally challenge requirements
  - Importance
  - Specific
- Don’t solve bigger problems than what you actually have
- KISS. Always!
- Automation testing worth its weight in gold!

© Fortinet Inc. All Rights Reserved.

## Testing Vs. Code Review

- Thank you!

center the application/ user experience on DB structure

flood with options

© Fortinet Inc. All Rights Reserved.

# The main two points:

- Frame the time spent on fix/feature
    - Back to the scratching board

© Fortinet Inc. All Rights Reserved.

## other

- The phoenix code
- Die fast (threads with pokemon catch)

- Make pattern.
- Not too specific
- How to recognize the pattern. When I encounter this.
  - If a, b and c  happens – this is what you do

- Tools, profiling…

© Fortinet Inc. All Rights Reserved.