DevOps or Death

(The title is silly)

Michele Finelli
m@biodec.com
BioDec

Index

Index

Cloud + Virtualization = Distributed Infrastructure

THANKS TO THE CLOUD AND TO THE VIRTUALIZATION technologies every company will need tools and techniques to deal with the complexities of a distributed infrastructure.

The **time frame** is the next *few* years, not decades.

The **alternative** is *giving up* controlling or *giving up* the advantages of those technologies.

Cloud + Virtualization = Distributed Infrastructure

THANKS TO THE CLOUD AND TO THE VIRTUALIZATION technologies every company will need tools and techniques to deal with the complexities of a distributed infrastructure.
The **time frame** is the next *few* years, not decades.
The **alternative** is *giving up* controlling or *giving up* the advantages of those technologies.

Cloud + Virtualization = Distributed Infrastructure

THANKS TO THE CLOUD AND TO THE VIRTUALIZATION technologies every company will need tools and techniques to deal with the complexities of a distributed infrastructure.
The **time frame** is the next *few* years, not decades.
The **alternative** is *giving up* controlling or *giving up* the advantages of those technologies.

A *large* distributed infrastructure

- Your infrastructure will be not only *distributed*, but also **large**: a very different scenario from that of traditional (monolithic) infrastructures.
- Different rules, different problems and different solutions.
- Hint: adapting to the **large** what happens in the **small** is not a good strategy . . .

A *large* distributed infrastructure

- Your infrastructure will be not only *distributed*, but also **large**: a very different scenario from that of traditional (monolithic) infrastructures.
- Different rules, different problems and different solutions.
- Hint: adapting to the **large** what happens in the **small** is not a good strategy . . .

A *large* distributed infrastructure

- Your infrastructure will be not only *distributed*, but also **large**: a very different scenario from that of traditional (monolithic) infrastructures.
- Different rules, different problems and different solutions.
- Hint: adapting to the **large** what happens in the **small** is not a good strategy . . .

A *large* distributed infrastructure

- Your infrastructure will be not only *distributed*, but also **large**: a very different scenario from that of traditional (monolithic) infrastructures.
- Different rules, different problems and different solutions.
- Hint: adapting to the **large** what happens in the **small** is not a good strategy ...

Index

Index

A time line*

At the very beginning . . . Patrick Debois, year 2007. He is doing a job that requires *hybrid* skills: both programmer and sysadmin.

Agile 2008 Andrew Shafer proposes a session on *"Agile Infrastructure"*, but exactly zero people show up.

June 2009 John Allspaw gives the talk *"10+ deploys per day: Dev & Ops cooperation at Flickr"*.

30-31 October 2009 The very first *DevOps Days* in Gent, Belgium.

*from a presentation by Damon Edwards on IT Revolutions.

A time line*

At the very beginning . . . Patrick Debois, year 2007. He is doing a job that requires *hybrid* skills: both programmer and sysadmin.

Agile 2008 Andrew Shafer proposes a session on *"Agile Infrastructure"*, but exactly zero people show up.

June 2009 John Allspaw gives the talk *"10+ deploys per day: Dev & Ops cooperation at Flickr"*.

30-31 October 2009 The very first *DevOps Days* in Gent, Belgium.

*from a presentation by Damon Edwards on IT Revolutions.

A time line*

At the very beginning . . . Patrick Debois, year 2007. He is doing a job that requires *hybrid* skills: both programmer and sysadmin.

Agile 2008 Andrew Shafer proposes a session on *"Agile Infrastructure"*, but exactly zero people show up.

June 2009 John Allspaw gives the talk *"10+ deploys per day: Dev & Ops cooperation at Flickr"*.

30-31 October 2009 The very first *DevOps Days* in Gent, Belgium.

*from a presentation by Damon Edwards on IT Revolutions.

A time line*

At the very beginning . . . Patrick Debois, year 2007. He is doing a job that requires *hybrid* skills: both programmer and sysadmin.

Agile 2008 Andrew Shafer proposes a session on *"Agile Infrastructure"*, but exactly zero people show up.

June 2009 John Allspaw gives the talk *"10+ deploys per day: Dev & Ops cooperation at Flickr"*.

30-31 October 2009 The very first *DevOps Days* in Gent, Belgium.

*from a presentation by Damon Edwards on IT Revolutions.

The early days

THE HASHTAG #DEVOPS is adopted. The *devops* topic — even if still undefined — gets debated in conferences and a new brand of community-driven meetings are organized all over the world.

More than **one hundred DevOpsDays** in eigth years, with thousands of participants.

The early days

THE HASHTAG #DEVOPS is adopted. The *devops* topic — even if still undefined — gets debated in conferences and a new brand of community-driven meetings are organized all over the world.
More than **one hundred DevOpsDays** in eigth years, with thousands of participants.

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

The early days

Attendees discuss about:

1. modern IT management,
2. techniques and tools to manage large infrastructure,
3. providing value to enterprise through faster delivery cycles, and faster deployment,
4. bridging the gap between developers and operations — does it ring a bell ?

All that is old is new again

Uncle Bob Martin says:

> *In 2001 a few of us met in hopes that we could agree on a simple statement that defines lightweight processes. We wrote a simple manifesto, and chose the name Agile. We had no idea how successful this idea would be. At that meeting, Kent Beck stated a prime goal: "**To heal the divide between business and development**".*

Yesterday

MARCH, 2011. Gartner Group publishes the report *"The Rise of a New IT Operations Support Model"*, where it is stated that by year **2015** the DevOps movement would have grown from *a niche movement for cloud companies* to **adoption in more than a fifth of *Global 2000* enterprises**.

Italy: what's happening ?

February, 2013 Florence, first Italian DevOps Meeting (Incontro DevOps Italia), 80+ people, *community driven*.

February, 2014 Bologna, second Italian DevOps Meeting, 120+ people: the *first sponsors*.

March 2015 Bologna, third Italian DevOps Meeting, 190+ people: first edition with *two parallel tracks*.

April 2016 Bologna, fourth Italian DevOps Meeting, 240+ people: first edition with *an international track*.

Italy: what's happening ?

February, 2013 Florence, first Italian DevOps Meeting (Incontro DevOps Italia), 80+ people, *community driven*.

February, 2014 Bologna, second Italian DevOps Meeting, 120+ people: the *first sponsors*.

March 2015 Bologna, third Italian DevOps Meeting, 190+ people: first edition with *two parallel tracks*.

April 2016 Bologna, fourth Italian DevOps Meeting, 240+ people: first edition with *an international track*.

Italy: what's happening ?

February, 2013 Florence, first Italian DevOps Meeting (Incontro DevOps Italia), 80+ people, *community driven*.

February, 2014 Bologna, second Italian DevOps Meeting, 120+ people: the *first sponsors*.

March 2015 Bologna, third Italian DevOps Meeting, 190+ people: first edition with *two parallel tracks*.

April 2016 Bologna, fourth Italian DevOps Meeting, 240+ people: first edition with *an international track*.

Italy: what's happening ?

February, 2013 Florence, first Italian DevOps Meeting (Incontro DevOps Italia), 80+ people, *community driven*.

February, 2014 Bologna, second Italian DevOps Meeting, 120+ people: the *first sponsors*.

March 2015 Bologna, third Italian DevOps Meeting, 190+ people: first edition with *two parallel tracks*.

April 2016 Bologna, fourth Italian DevOps Meeting, 240+ people: first edition with *an international track*.

Italy: what's happening ?

February, 2013 Florence, first Italian DevOps Meeting (Incontro DevOps Italia), 80+ people, *community driven*.

February, 2014 Bologna, second Italian DevOps Meeting, 120+ people: the *first sponsors*.

March 2015 Bologna, third Italian DevOps Meeting, 190+ people: first edition with *two parallel tracks*.

April 2016 Bologna, fourth Italian DevOps Meeting, 240+ people: first edition with *an international track*.

See you next year

When 7th and 8th March 2017.

Where Bologna.

What Fifth Italian DevOps Meeting: **one day of workshops, and and one day of conference.**

How http://www.incontrodevops.it/

See you next year

When 7th and 8th March 2017.

Where Bologna.

What Fifth Italian DevOps Meeting: **one day of workshops, and and one day of conference.**

How `http://www.incontrodevops.it/`

See you next year

When 7th and 8th March 2017.

Where Bologna.

What Fifth Italian DevOps Meeting: **one day of workshops, and and one day of conference.**

How http://www.incontrodevops.it/

See you next year

When 7th and 8th March 2017.

Where Bologna.

What Fifth Italian DevOps Meeting: **one day of workshops, and and one day of conference**.

How http://www.incontrodevops.it/

See you next year

When 7th and 8th March 2017.

Where Bologna.

What Fifth Italian DevOps Meeting: **one day of workshops, and and one day of conference**.

How `http://www.incontrodevops.it/`

Index

So, what is this DevOps thing ?

MY VERY PERSONAL OPINION is that DevOps is a *pot-pourri* of different things, ideas, techniques and practices. I would say that it is a blend of:

1. agile *stuff*,
2. lean methodologies,
3. some characteristics of the free software communities: openness, sharing, open standards,
4. and probably something else.

So, what is this DevOps thing ?

MY VERY PERSONAL OPINION is that DevOps is a *pot-pourri* of different things, ideas, techniques and practices. I would say that it is a blend of:

1. agile *stuff*,
2. lean methodologies,
3. some characteristics of the free software communities: openness, sharing, open standards,
4. and probably something else.

So, what is this DevOps thing ?

MY VERY PERSONAL OPINION is that DevOps is a *pot-pourri* of different things, ideas, techniques and practices. I would say that it is a blend of:

1. agile *stuff*,
2. lean methodologies,
3. some characteristics of the free software communities: openness, sharing, open standards,
4. and probably something else.

So, what is this DevOps thing ?

MY VERY PERSONAL OPINION is that DevOps is a *pot-pourri* of different things, ideas, techniques and practices. I would say that it is a blend of:

1. agile *stuff*,
2. lean methodologies,
3. some characteristics of the free software communities: openness, sharing, open standards,
4. and probably something else.

So, what is this DevOps thing ?

MY VERY PERSONAL OPINION is that DevOps is a *pot-pourri* of different things, ideas, techniques and practices. I would say that it is a blend of:

1. agile *stuff*,
2. lean methodologies,
3. some characteristics of the free software communities: openness, sharing, open standards,
4. and probably something else.

. . . and what it is not

Defining DevOps by *negation*. That part is easier:

1. it is *not* a certification,
2. it is *not* a job title,
3. it is *not* a tool nor a software.

. . . and what it is not

Defining DevOps by *negation*. That part is easier:

1. it is *not* a certification,
2. it is *not* a job title,
3. it is *not* a tool nor a software.

. . . and what it is not

Defining DevOps by *negation*. That part is easier:

1. it is *not* a certification,
2. it is *not* a job title,
3. it is *not* a tool nor a software.

. . . and what it is not

Defining DevOps by *negation*. That part is easier:

1. it is *not* a certification,
2. it is *not* a job title,
3. it is *not* a tool nor a software.

An acronym: CAMS

C *culture*

A *automate*

M *measure*

S *share*

An acronym: CAMS

C *culture*

A *automate*

M *measure*

S *share*

An acronym: CAMS

C *culture*

A *automate*

M *measure*

S *share*

An acronym: CAMS

C *culture*

A *automate*

M *measure*

S *share*

An acronym: CAMS

C *culture*

A *automate*

M *measure*

S *share*

Culture

1 CREATE A CULTURE of collaboration. The first issue is the harder to get in practice, but it is probably the most important.

> *People and process first. If you don't have culture, all automation attempts will be fruitless. (John Willis)*

Automate

2 AUTOMATE everything. Let any task that can be done with software, be done by a program: write it, deploy it and run it.

All software is born equal under the sun: a *system* program is not an excuse for sloppy practices, lack of quality or misfeasance.

Corollary: *sysadmin* is not an insult.

Automate

2 AUTOMATE everything. Let any task that can be done with software, be done by a program: write it, deploy it and run it.
All software is born equal under the sun: a *system* program is not an excuse for sloppy practices, lack of quality or misfeasance.
Corollary: *sysadmin* is not an insult.

Automate

2 AUTOMATE everything. Let any task that can be done with software, be done by a program: write it, deploy it and run it.
All software is born equal under the sun: a *system* program is not an excuse for sloppy practices, lack of quality or misfeasance.
Corollary: *sysadmin* is not an insult.

Infrastructure as code

Since only software determines what can be done, it means that also the *lack of it* defines what *cannot be done*, and in particular it is not acceptable to have:

1. hand-made configurations (*i.e.* snowflakes servers),
2. things that happens clicking on interfaces, with no versioning or change management in place,
3. people (*a.k.a.* consultants) that come, cast a spell and run away with money.

Infrastructure as code

Since only software determines what can be done, it means that also the *lack of it* defines what *cannot be done*, and in particular it is not acceptable to have:

1. hand-made configurations (*i.e.* snowflakes servers),
2. things that happens clicking on interfaces, with no versioning or change management in place,
3. people (*a.k.a.* consultants) that come, cast a spell and run away with money.

Infrastructure as code

Since only software determines what can be done, it means that also the *lack of it* defines what *cannot be done*, and in particular it is not acceptable to have:

1. hand-made configurations (*i.e.* snowflakes servers),
2. things that happens clicking on interfaces, with no versioning or change management in place,
3. people (*a.k.a.* consultants) that come, cast a spell and run away with money.

Infrastructure as code

Since only software determines what can be done, it means that also the *lack of it* defines what *cannot be done*, and in particular it is not acceptable to have:

1. hand-made configurations (*i.e.* snowflakes servers),
2. things that happens clicking on interfaces, with no versioning or change management in place,
3. people (*a.k.a.* consultants) that come, cast a spell and run away with money.

Measure everything

3 MEASURE all the parts of the infrastructure. Monitoring is not a new idea, and it has obviously not been invented by the DevOps community: the novelty of the DevOps approach is in considering monitoring *as a whole*: systems, applications, network. **Everything has to be available for anybody involved**.

Measure everything

The traditional approach to monitoring consists of some system management tool, **usually just for the system administrators**, tracking server resources or hardware performance data.
Trouble arise since that tool is usually decoupled from an **ad hoc solution devised for the applications, by the application developer themselves**.

Share

4 SHARE a project outcome, an objective, practices, techniques, tools among different groups that have different roles and responsibilities.

> *Sharing is the loopback in the CAMS cycle. Creating a culture where people share ideas and problems is critical. (John Willis)*

Wrapping up

There is a chain of implications, and if you accept the premises the conclusion is inescapable:

1. if only the code defines the infrastructure,
2. and every action on the infrastructure has to be automated (that means: translated into code),
3. then the only way of determining an effect on the infrastructure is by programming,
4. and this means that **you are programmer**, willing or not, independently of your job title.

Wrapping up

There is a chain of implications, and if you accept the premises the conclusion is inescapable:

1. if only the code defines the infrastructure,
2. and every action on the infrastructure has to be automated (that means: translated into code),
3. then the only way of determining an effect on the infrastructure is by programming,
4. and this means that **you are programmer**, willing or not, independently of your job title.

Wrapping up

There is a chain of implications, and if you accept the premises the conclusion is inescapable:

1. if only the code defines the infrastructure,
2. and every action on the infrastructure has to be automated (that means: translated into code),
3. then the only way of determining an effect on the infrastructure is by programming,
4. and this means that **you are programmer**, willing or not, independently of your job title.

Wrapping up

There is a chain of implications, and if you accept the premises the conclusion is inescapable:

1. if only the code defines the infrastructure,
2. and every action on the infrastructure has to be automated (that means: translated into code),
3. then the only way of determining an effect on the infrastructure is by programming,
4. and this means that **you are programmer**, willing or not, independently of your job title.

Wrapping up

There is a chain of implications, and if you accept the premises the conclusion is inescapable:

1. if only the code defines the infrastructure,
2. and every action on the infrastructure has to be automated (that means: translated into code),
3. then the only way of determining an effect on the infrastructure is by programming,
4. and this means that **you are programmer**, willing or not, independently of your job title.

Index

Index

Technique #1: repeatable setup

- Setting up a new server or a new service should be **repeatable task**: *i.e.* it should not require manual operations or manual configurations.
- Do it !
- Ask your providers to work that way.

Technique #1: repeatable setup

- Setting up a new server or a new service should be **repeatable task**: *i.e.* it should not require manual operations or manual configurations.
- Do it !
- Ask your providers to work that way.

Technique #1: repeatable setup

- Setting up a new server or a new service should be **repeatable task**: *i.e.* it should not require manual operations or manual configurations.
- Do it !
- Ask your providers to work that way.

Technique #1: repeatable setup

- Setting up a new server or a new service should be **repeatable task**: *i.e.* it should not require manual operations or manual configurations.
- Do it !
- Ask your providers to work that way.

Advantages

- Setting up environments for different needs is possible (testing, staging, production, development from different providers, etcetera).
- The setup procedure itself can be subject of further validation.
- Speed: setting up the machinery is slow, but running the task is fast.

Advantages

- Setting up environments for different needs is possible (testing, staging, production, development from different providers, etcetera).
- The setup procedure itself can be subject of further validation.
- Speed: setting up the machinery is slow, but running the task is fast.

Advantages

- Setting up environments for different needs is possible (testing, staging, production, development from different providers, etcetera).
- The setup procedure itself can be subject of further validation.
- Speed: setting up the machinery is slow, but running the task is fast.

Advantages

- Setting up environments for different needs is possible (testing, staging, production, development from different providers, etcetera).
- The setup procedure itself can be subject of further validation.
- Speed: setting up the machinery is slow, but running the task is fast.

Costs

- Time: automating procedures takes **a lot** of time.
- Skills: a different mindset is needed to manage the infrastructure as code — no more *“Let me connect to the server and change some settings”*.

Costs

- Time: automating procedures takes **a lot** of time.
- Skills: a different mindset is needed to manage the infrastructure as code — no more *“Let me connect to the server and change some settings”*.

Costs

- Time: automating procedures takes **a lot** of time.
- Skills: a different mindset is needed to manage the infrastructure as code — no more *“Let me connect to the server and change some settings”*.

Tool #1: configuration management system

- Keep everything in a configuration management system.
- File systems are bad; repositories are good.
- Ask your suppliers to do the same (if they say they can't, they are lying). **Offer to train them.**

Tool #1: configuration management system

- Keep everything in a configuration management system.
- File systems are bad; repositories are good.
- Ask your suppliers to do the same (if they say they can't, they are lying). **Offer to train them.**

Tool #1: configuration management system

- Keep everything in a configuration management system.
- File systems are bad; repositories are good.
- Ask your suppliers to do the same (if they say they can't, they are lying). **Offer to train them.**

Tool #1: configuration management system

- Keep everything in a configuration management system.
- File systems are bad; repositories are good.
- Ask your suppliers to do the same (if they say they can't, they are lying). **Offer to train them.**

Advantages

- Changes are traceable. Accountability.
- Recovering from mistakes simply by *undo* changes.
- Setting *releases* of artifacts, to mark the state of a point in time.

Advantages

- Changes are traceable. Accountability.
- Recovering from mistakes simply by *undo* changes.
- Setting *releases* of artifacts, to mark the state of a point in time.

Advantages

- Changes are traceable. Accountability.
- Recovering from mistakes simply by *undo* changes.
- Setting *releases* of artifacts, to mark the state of a point in time.

Advantages

- Changes are traceable. Accountability.
- Recovering from mistakes simply by *undo* changes.
- Setting *releases* of artifacts, to mark the state of a point in time.

Costs

- Discipline.

Costs

- Discipline.

Technique #2: testing

- Define test before or together with the implementation of the software solution.
- Do not accept anything that has not been properly tested. How do you verify that tests do indeed:
  1. exists;
  2. are passed by the software you are going to deploy;

  ?
- See technique #1 (repeatability) and tool #1 (configuration management of all artifacts).

Technique #2: testing

- Define test before or together with the implementation of the software solution.
- Do not accept anything that has not been properly tested. How do you verify that tests do indeed:
  1. exists;
  2. are passed by the software you are going to deploy;

  ?
- See technique #1 (repeatability) and tool #1 (configuration management of all artifacts).

Technique #2: testing

- Define test before or together with the implementation of the software solution.
- Do not accept anything that has not been properly tested. How do you verify that tests do indeed:
  1. exists;
  2. are passed by the software you are going to deploy;

  ?
- See technique #1 (repeatability) and tool #1 (configuration management of all artifacts).

Technique #2: testing

- Define test before or together with the implementation of the software solution.
- Do not accept anything that has not been properly tested. How do you verify that tests do indeed:
  1. exists;
  2. are passed by the software you are going to deploy;

  ?
- See technique #1 (repeatability) and tool #1 (configuration management of all artifacts).

Advantages

- The probability of broken code is lesser and lesser over time.
- The necessity of rollbacks or cut-corner patches to fix unexpected problems is lesser and lesser.
- Quality.

Advantages

- The probability of broken code is lesser and lesser over time.
- The necessity of rollbacks or cut-corner patches to fix unexpected problems is lesser and lesser.
- Quality.

Advantages

- The probability of broken code is lesser and lesser over time.
- The necessity of rollbacks or cut-corner patches to fix unexpected problems is lesser and lesser.
- Quality.

Advantages

- The probability of broken code is lesser and lesser over time.
- The necessity of rollbacks or cut-corner patches to fix unexpected problems is lesser and lesser.
- Quality.

Costs

- Skills: tests must be conceived before or at least together with the solution, non as an afterthought.
- Discipline: Keeping the course under time pressure is hard.
- Use those levers at your advantage as a way of selecting and evaluating your supply chain.

Costs

- Skills: tests must be conceived before or at least together with the solution, non as an afterthought.
- Discipline: Keeping the course under time pressure is hard.
- Use those levers at your advantage as a way of selecting and evaluating your supply chain.

Costs

- Skills: tests must be conceived before or at least together with the solution, non as an afterthought.
- Discipline: Keeping the course under time pressure is hard.
- Use those levers at your advantage as a way of selecting and evaluating your supply chain.

Costs

- Skills: tests must be conceived before or at least together with the solution, non as an afterthought.
- Discipline: Keeping the course under time pressure is hard.
- Use those levers at your advantage as a way of selecting and evaluating your supply chain.

Tool #2: continuous integration

- Any change to the system triggers a build and all the relevant tests are run: changes that break the build are rejected.
- There are free software tools that can be used to make a *pipeline* with little effort.

Tool #2: continuous integration

- Any change to the system triggers a build and all the relevant tests are run: changes that break the build are rejected.
- There are free software tools that can be used to make a *pipeline* with little effort.

Tool #2: continuous integration

- Any change to the system triggers a build and all the relevant tests are run: changes that break the build are rejected.
- There are free software tools that can be used to make a *pipeline* with little effort.

Advantages

- Linking together the code and the tests as whole: untested code is broken code.
- Quality.

Advantages

- Linking together the code and the tests as whole: untested code is broken code.
- Quality.

Advantages

- Linking together the code and the tests as whole: untested code is broken code.
- Quality.

Costs

- As for testing: skills and discipline.
- Some resources (not too many).
- Time: cutting corners saves times.

Costs

- As for testing: skills and discipline.
- Some resources (not too many).
- Time: cutting corners saves times.

Costs

- As for testing: skills and discipline.
- Some resources (not too many).
- Time: cutting corners saves times.

Costs

- As for testing: skills and discipline.
- Some resources (not too many).
- Time: cutting corners saves times.

Technique #3: visualize information

- Once you have a complex, distributed and large infrastructure, the only way to know that it working is having either *key process indicators* (KPIs) and the system that checks them.
- I KPIs are hidden, they are worthless: relevant information must be easily accessible.

Technique #3: visualize information

- Once you have a complex, distributed and large infrastructure, the only way to know that it working is having either *key process indicators* (KPIs) and the system that checks them.
- I KPIs are hidden, they are worthless: relevant information must be easily accessible.

Technique #3: visualize information

- Once you have a complex, distributed and large infrastructure, the only way to know that it working is having either *key process indicators* (KPIs) and the system that checks them.
- I KPIs are hidden, they are worthless: relevant information must be easily accessible.

Advantages

- Shorter feedback loops.
- Better understanding of the system architecture.
- Help in planning and evolving the system.

Advantages

- Shorter feedback loops.
- Better understanding of the system architecture.
- Help in planning and evolving the system.

Advantages

- Shorter feedback loops.
- Better understanding of the system architecture.
- Help in planning and evolving the system.

Advantages

- Shorter feedback loops.
- Better understanding of the system architecture.
- Help in planning and evolving the system.

Costs

- Too much openness in some companies is counterproductive.
- (Change company).
- (Positive form: change supplier or provider).

Costs

- Too much openness in some companies is counterproductive.
- (Change company).
- (Positive form: change supplier or provider).

Costs

- Too much openness in some companies is counterproductive.
- (Change company).
- (Positive form: change supplier or provider).

Costs

- Too much openness in some companies is counterproductive.
- (Change company).
- (Positive form: change supplier or provider).

Tool #3: a *modern* monitoring system

- Setup a modern monitoring system to gather logs, show metrics and manage alerts.
- There are many really good free software solutions.
- Implement it and keep it up to date.

Tool #3: a *modern* monitoring system

- Setup a modern monitoring system to gather logs, show metrics and manage alerts.
- There are many really good free software solutions.
- Implement it and keep it up to date.

Tool #3: a *modern* monitoring system

- Setup a modern monitoring system to gather logs, show metrics and manage alerts.
- There are many really good free software solutions.
- Implement it and keep it up to date.

Tool #3: a *modern* monitoring system

- Setup a modern monitoring system to gather logs, show metrics and manage alerts.
- There are many really good free software solutions.
- Implement it and keep it up to date.

Advantages

- It is a solution to the problem of building a system that *visualizes* the infrastructure.
- It helps in building repeatable processes to evolve the infrastructure, address incidents, etcetera.
- Quantitative information is often necessary to prove accountability and quality.

Advantages

- It is a solution to the problem of building a system that *visualizes* the infrastructure.
- It helps in building repeatable processes to evolve the infrastructure, address incidents, etcetera.
- Quantitative information is often necessary to prove accountability and quality.

Advantages

- It is a solution to the problem of building a system that *visualizes* the infrastructure.
- It helps in building repeatable processes to evolve the infrastructure, address incidents, etcetera.
- Quantitative information is often necessary to prove accountability and quality.

Advantages

- It is a solution to the problem of building a system that *visualizes* the infrastructure.
- It helps in building repeatable processes to evolve the infrastructure, address incidents, etcetera.
- Quantitative information is often necessary to prove accountability and quality.

Costs

- Time: building a *fairly complete* monitoring system takes a lot of time and energy.
- Often the process shows all the dark spots, the corners that were cut, etcetera.
- Once built, the monitoring system must be evolved together with the main infrastructure: in fact it is another more complexity to manage.

Costs

- Time: building a *fairly complete* monitoring system takes a lot of time and energy.
- Often the process shows all the dark spots, the corners that were cut, etcetera.
- Once built, the monitoring system must be evolved together with the main infrastructure: in fact it is another more complexity to manage.

Costs

- Time: building a *fairly complete* monitoring system takes a lot of time and energy.
- Often the process shows all the dark spots, the corners that were cut, etcetera.
- Once built, the monitoring system must be evolved together with the main infrastructure: in fact it is another more complexity to manage.

Costs

- Time: building a *fairly complete* monitoring system takes a lot of time and energy.
- Often the process shows all the dark spots, the corners that were cut, etcetera.
- Once built, the monitoring system must be evolved together with the main infrastructure: in fact it is another more complexity to manage.

Principle #1: avoid lock-in at all costs

- A requirement of the above initiatives is the capability to *operate* on the infrastructure itself.
- Being cut-off because of licensing issues, legacy systems and so on is a major stopover.
- There are techniques to *fence* the parts of the infrastructure that cannot be managed as the others.

Principle #1: avoid lock-in at all costs

- A requirement of the above initiatives is the capability to *operate* on the infrastructure itself.
- Being cut-off because of licensing issues, legacy systems and so on is a major stopover.
- There are techniques to *fence* the parts of the infrastructure that cannot be managed as the others.

Principle #1: avoid lock-in at all costs

- A requirement of the above initiatives is the capability to *operate* on the infrastructure itself.
- Being cut-off because of licensing issues, legacy systems and so on is a major stopover.
- There are techniques to *fence* the parts of the infrastructure that cannot be managed as the others.

Principle #1: avoid lock-in at all costs

- A requirement of the above initiatives is the capability to *operate* on the infrastructure itself.
- Being cut-off because of licensing issues, legacy systems and so on is a major stopover.
- There are techniques to *fence* the parts of the infrastructure that cannot be managed as the others.

Advantages

- Being able to fully harvest the benefits of other initiatives.
- Better control.
- Better flexibility and ability to respond to changes.

Advantages

- Being able to fully harvest the benefits of other initiatives.
- Better control.
- Better flexibility and ability to respond to changes.

Advantages

- Being able to fully harvest the benefits of other initiatives.
- Better control.
- Better flexibility and ability to respond to changes.

Advantages

- Being able to fully harvest the benefits of other initiatives.
- Better control.
- Better flexibility and ability to respond to changes.

Costs

- Walled gardens are tempting: often a solution that guarantees more freedom is more expensive (to build, to manage, to understand, to use, etcetera).
- Sometimes you will have to sail upstream, and face also organizational issues, policies, commercial agreements and so on.

Costs

- Walled gardens are tempting: often a solution that guarantees more freedom is more expensive (to build, to manage, to understand, to use, etcetera).
- Sometimes you will have to sail upstream, and face also organizational issues, policies, commercial agreements and so on.

Costs

- Walled gardens are tempting: often a solution that guarantees more freedom is more expensive (to build, to manage, to understand, to use, etcetera).
- Sometimes you will have to sail upstream, and face also organizational issues, policies, commercial agreements and so on.

Index

Fostering collaboration through a common core of tools and techniques

- Configuration management, monitoring and continuous integration are **amazingly useful** and more and more **mandatory** in the next future.
- They are components that — because of their nature — are **cross-functional**, since they integrate application (*Dev*) and system (*Ops*) information. Data integration is an **enabler** for stuff (*i.e.* bug free software, continuous deployment, etcetera) that would otherwise be **impossible** or too expensive to achieve.

Fostering collaboration through a common core of tools and techniques

- Configuration management, monitoring and continuous integration are **amazingly useful** and more and more **mandatory** in the next future.
- They are components that — because of their nature — are **cross-functional**, since they integrate application (*Dev*) and system (*Ops*) information. Data integration is an **enabler** for stuff (*i.e.* bug free software, continuous deployment, etcetera) that would otherwise be **impossible** or too expensive to achieve.

Fostering collaboration through a common core of tools and techniques

- Configuration management, monitoring and continuous integration are **amazingly useful** and more and more **mandatory** in the next future.
- They are components that — because of their nature — are **cross-functional**, since they integrate application (*Dev*) and system (*Ops*) information. Data integration is an **enabler** for stuff (*i.e.* bug free software, continuous deployment, etcetera) that would otherwise be **impossible** or too expensive to achieve.

Fostering collaboration through a common core of tools and techniques

- Configuration management, monitoring and continuous integration are **amazingly useful** and more and more **mandatory** in the next future.
- They are components that — because of their nature — are **cross-functional**, since they integrate application (*Dev*) and system (*Ops*) information. Data integration is an **enabler** for stuff (*i.e.* bug free software, continuous deployment, etcetera) that would otherwise be **impossible** or too expensive to achieve.

Fostering collaboration through a common core of tools and techniques

- Configuration management, monitoring and continuous integration are **amazingly useful** and more and more **mandatory** in the next future.
- They are components that — because of their nature — are **cross-functional**, since they integrate application (*Dev*) and system (*Ops*) information. Data integration is an **enabler** for stuff (*i.e.* bug free software, continuous deployment, etcetera) that would otherwise be **impossible** or too expensive to achieve.

Fostering collaboration through a common core of tools and techniques

- There are modern tools in the free software world that allow companies to build **modular** solutions **that can evolve**.
- Do not build or buy a **monolithic** solution !
- If you just do what everybody else is doing in the world of **high-performance companies**, you will have a robust solution to evolve your IT infrastructure and you will have enabled your teams **to operate as a single team**, and to bridge the divisions among different business areas.

Fostering collaboration through a common core of tools and techniques

- There are modern tools in the free software world that allow companies to build **modular** solutions **that can evolve**.
- Do not build or buy a **monolithic** solution !
- If you just do what everybody else is doing in the world of **high-performance companies**, you will have a robust solution to evolve your IT infrastructure and you will have enabled your teams **to operate as a single team**, and to bridge the divisions among different business areas.

Fostering collaboration through a common core of tools and techniques

- There are modern tools in the free software world that allow companies to build **modular** solutions **that can evolve**.
- Do not build or buy a **monolithic** solution !
- If you just do what everybody else is doing in the world of **high-performance companies**, you will have a robust solution to evolve your IT infrastructure and you will have enabled your teams **to operate as a single team**, and to bridge the divisions among different business areas.

Fostering collaboration through a common core of tools and techniques

- There are modern tools in the free software world that allow companies to build **modular** solutions **that can evolve**.
- Do not build or buy a **monolithic** solution !
- If you just do what everybody else is doing in the world of **high-performance companies**, you will have a robust solution to evolve your IT infrastructure and you will have enabled your teams **to operate as a single team**, and to bridge the divisions among different business areas.

Fostering collaboration through a common core of tools and techniques

- There are modern tools in the free software world that allow companies to build **modular** solutions **that can evolve**.
- Do not build or buy a **monolithic** solution !
- If you just do what everybody else is doing in the world of **high-performance companies**, you will have a robust solution to evolve your IT infrastructure and you will have enabled your teams **to operate as a single team**, and to bridge the divisions among different business areas.

Thanks & see you soon . . .

Thanks for participating !

IDI2017 Incontro DevOps Italia 2017 !!

More news on BioDec's blog at http://blog.biodec.com/
or at the event web site
http://www.incontrodevops.it/

*license of the slides:
http://creativecommons.org/licenses/by-sa/3.0/

Thanks & see you soon . . .

Thanks for participating !

IDI2017 Incontro DevOps Italia 2017 !!

More news on BioDec's blog at http://blog.biodec.com/
or at the event web site
http://www.incontrodevops.it/

*license of the slides:
http://creativecommons.org/licenses/by-sa/3.0/

Thanks & see you soon . . .

Thanks for participating !

IDI2017 Incontro DevOps Italia 2017 !!

More news on BioDec's blog at http://blog.biodec.com/
or at the event web site
http://www.incontrodevops.it/

*license of the slides:
http://creativecommons.org/licenses/by-sa/3.0/

Thanks & see you soon . . .

Thanks	for participating !
IDI2017	Incontro DevOps Italia 2017 !!
More news	on BioDec's blog at http://blog.biodec.com/ or at the event web site http://www.incontrodevops.it/

*license of the slides:
http://creativecommons.org/licenses/by-sa/3.0/