Big CI from scratch

Or how I stopped worrying and started to love the automatic test

March 24, 2021

NEXER

Disclaimer:

- Layman's experience from the trenches
- AKA: *Davids opinion considered harmful!*

NEXER

Disclaimer:

- Layman's experience from the trenches
- AKA: *Davids opinion considered harmful!*

NEXER

Disclaimer:

- Layman's experience from the trenches
- AKA: *Davids opinion considered harmful!*

NEXER

Ericsson RBS

NEXER

Ericsson RBS

- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Ericsson RBS

- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Ericsson RBS

- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Ericsson RBS

- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Ericsson RBS

- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Ericsson RBS

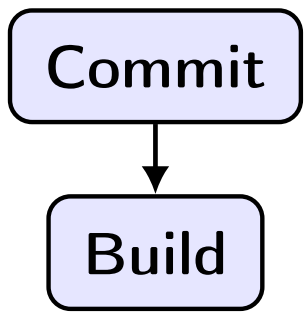
- Thousands of developers
- All developing for Radio Base Station
  - Different sub-organizations, different responsibilities
  - Nexer, one sub area $\approx$ couple hundred developers
- Gerrit / Git / Jenkins / Jira / (Eiffel)
- + in-house tools

NEXER

Basic CI

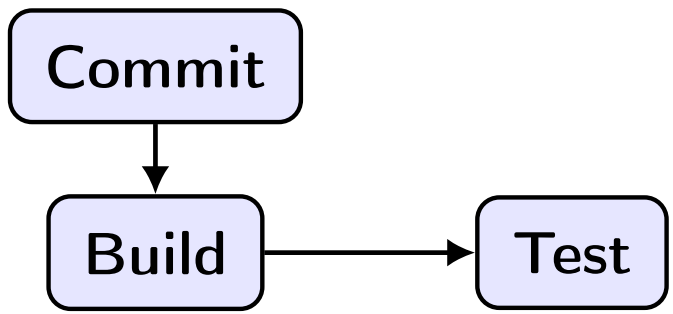
NEXXER

Basic CI

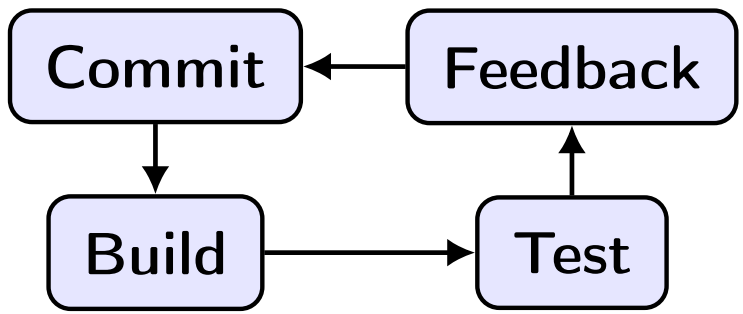


NEXXER

Basic CI



Basic CI



NEXXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

Big CI Problems

- Test scope
  - Can we run all tests?
  - Where should tests run?
  - Are all tests passing?
- Tracking
  - Where is my commit?
  - Is my commit ok?
- Intermittency
  - Lots of tests + intermittent tests $\equiv$ no flow
- Lead time
  - Feedback loop
- Many developers
  - $\rightarrow$ Many Bottlenecks
  - Dependencies (expected and unexpected!)

NEXER

CI from the CI Flow Plumber's point of view

- Modularization
- Logging

- Non-exhaustive list!
  - Speed
  - Stability
  - Reproducibility
  - Scalability
  - ...

NEXER

CI from the CI Flow Plumber's point of view

- Modularization
- Logging

- Non-exhaustive list!
  - Speed
  - Stability
  - Reproducibility
  - Scalability
  - ...

NEXXER

CI from the CI Flow Plumber's point of view

- Modularization
- Logging

- Non-exhaustive list!
  - Speed
  - Stability
  - Reproducibility
  - Scalability
  - ...

NEXXER

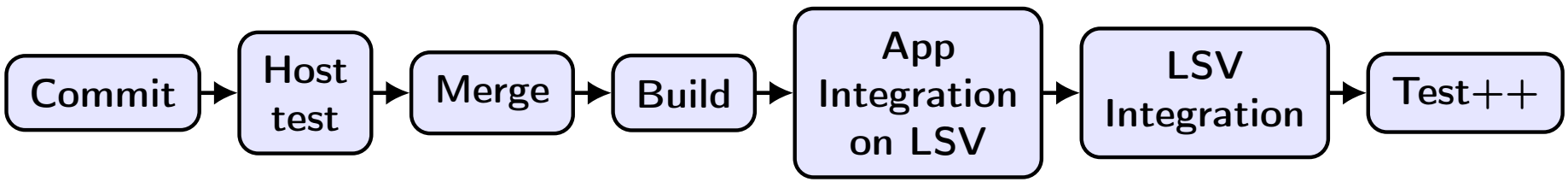
CI from the CI Flow Plumber's point of view

- Modularization
- Logging

- Non-exhaustive list!
  - Speed
  - Stability
  - Reproducibility
  - Scalability
  - ...

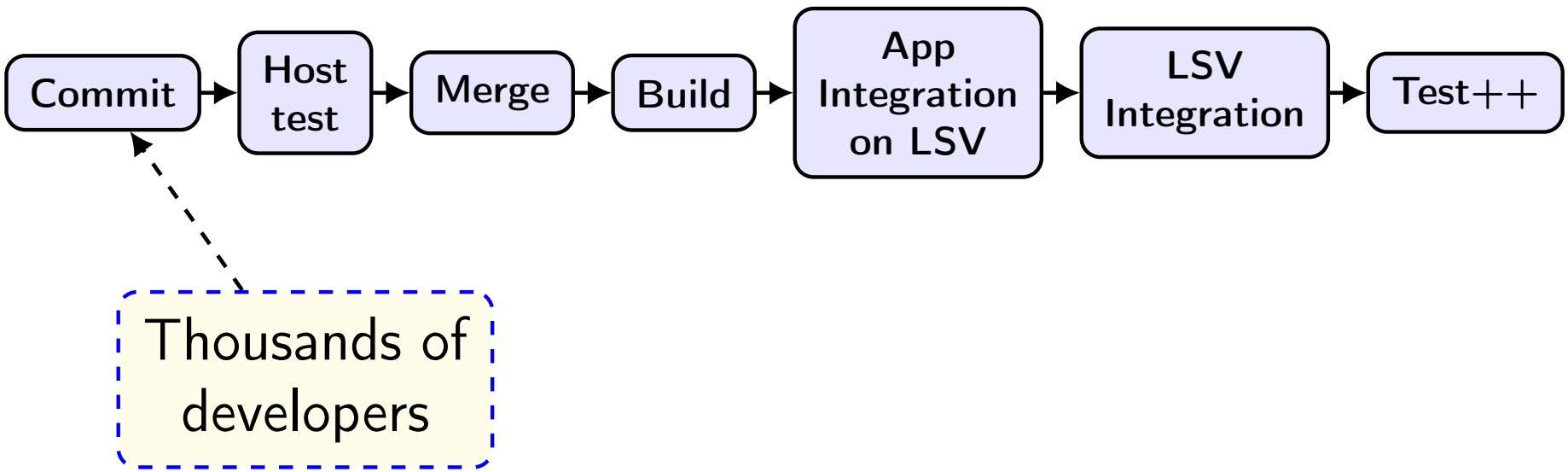
NEXER

Modularization



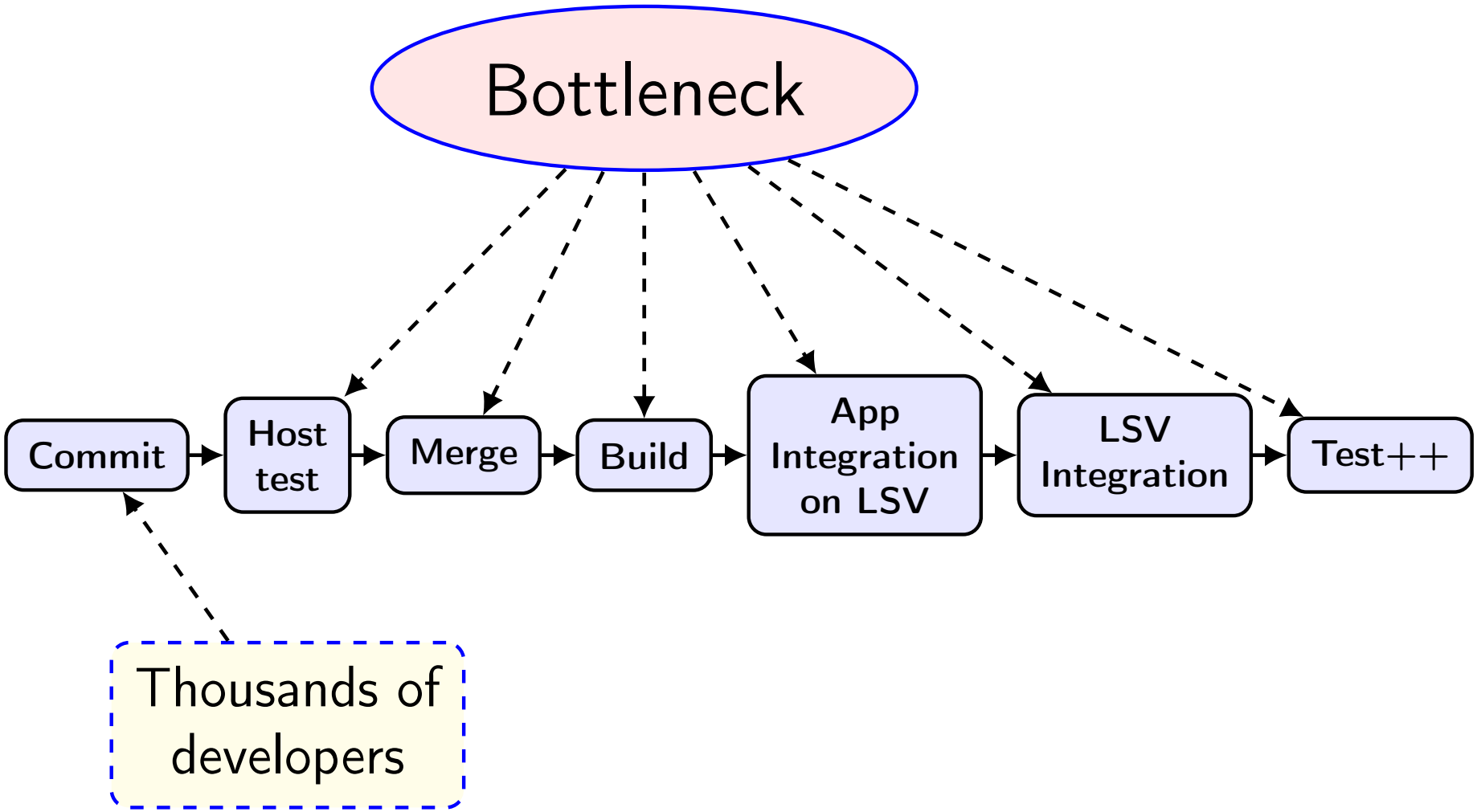
NEXXER

Modularization



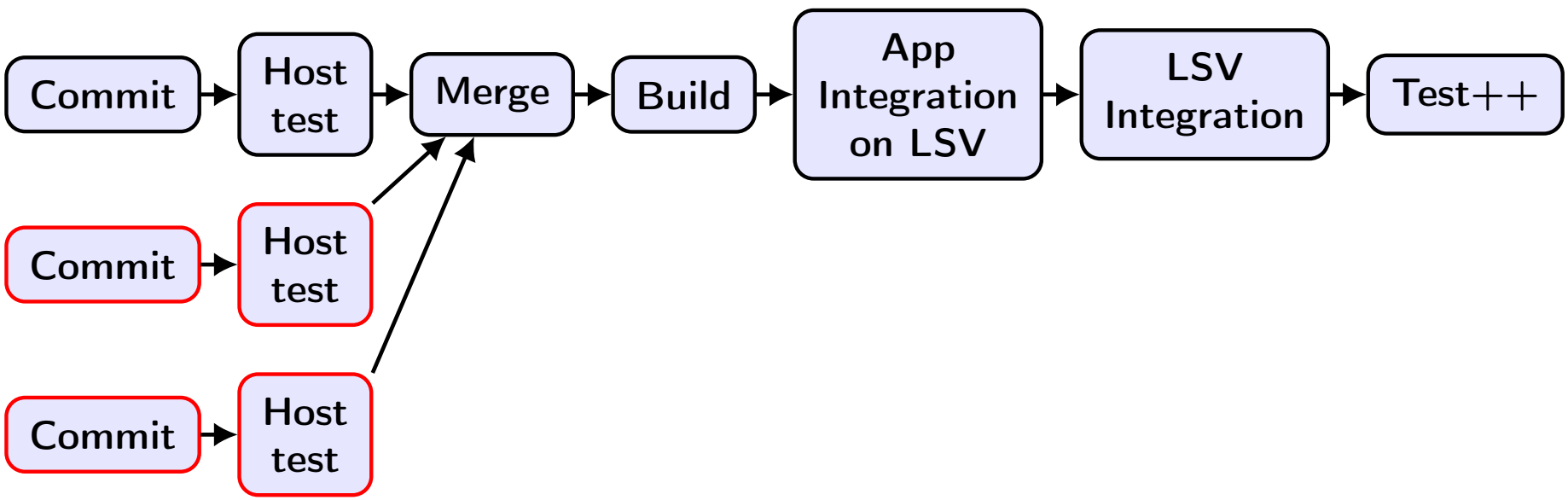
NEXER

Modularization



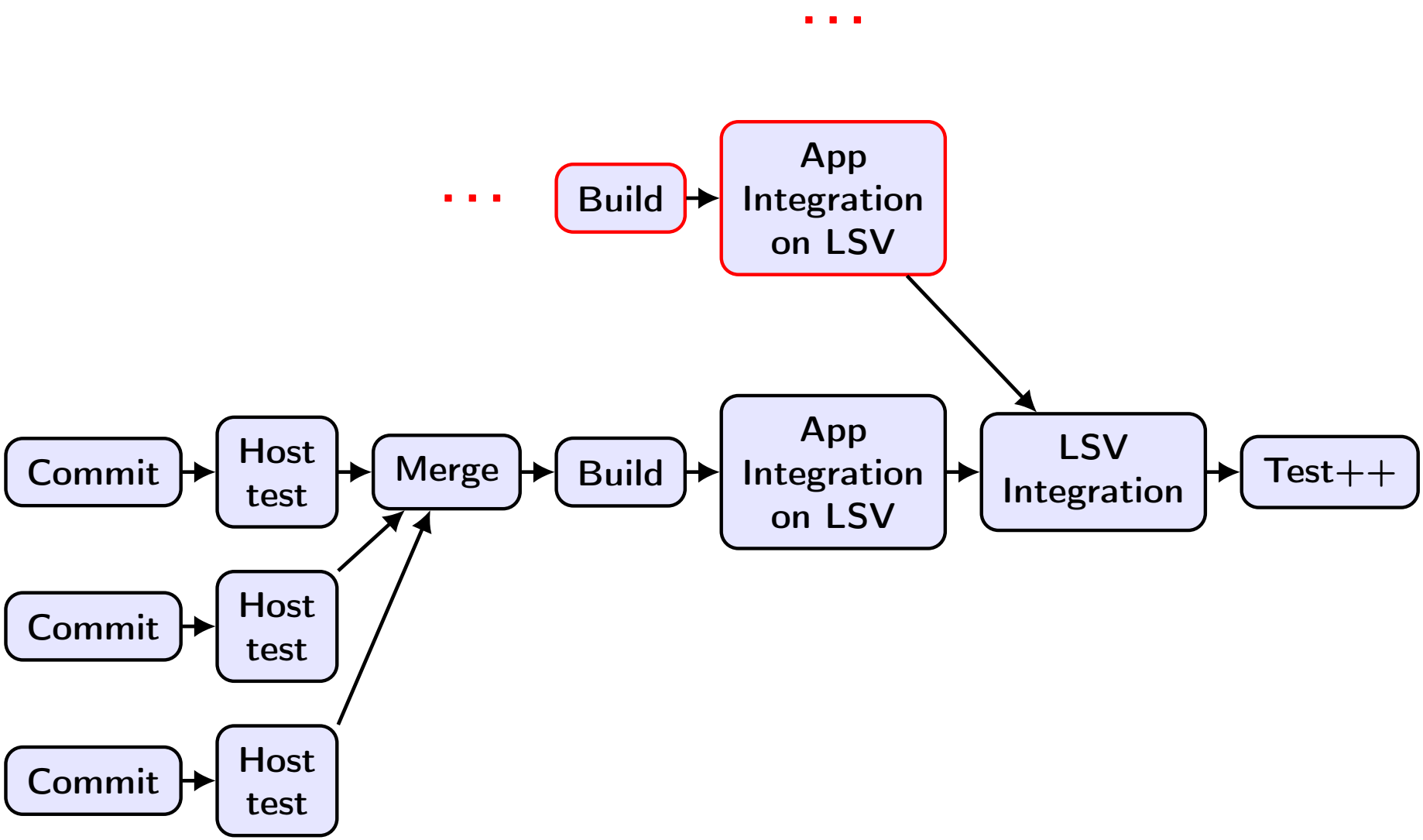
NEXER

Modularization

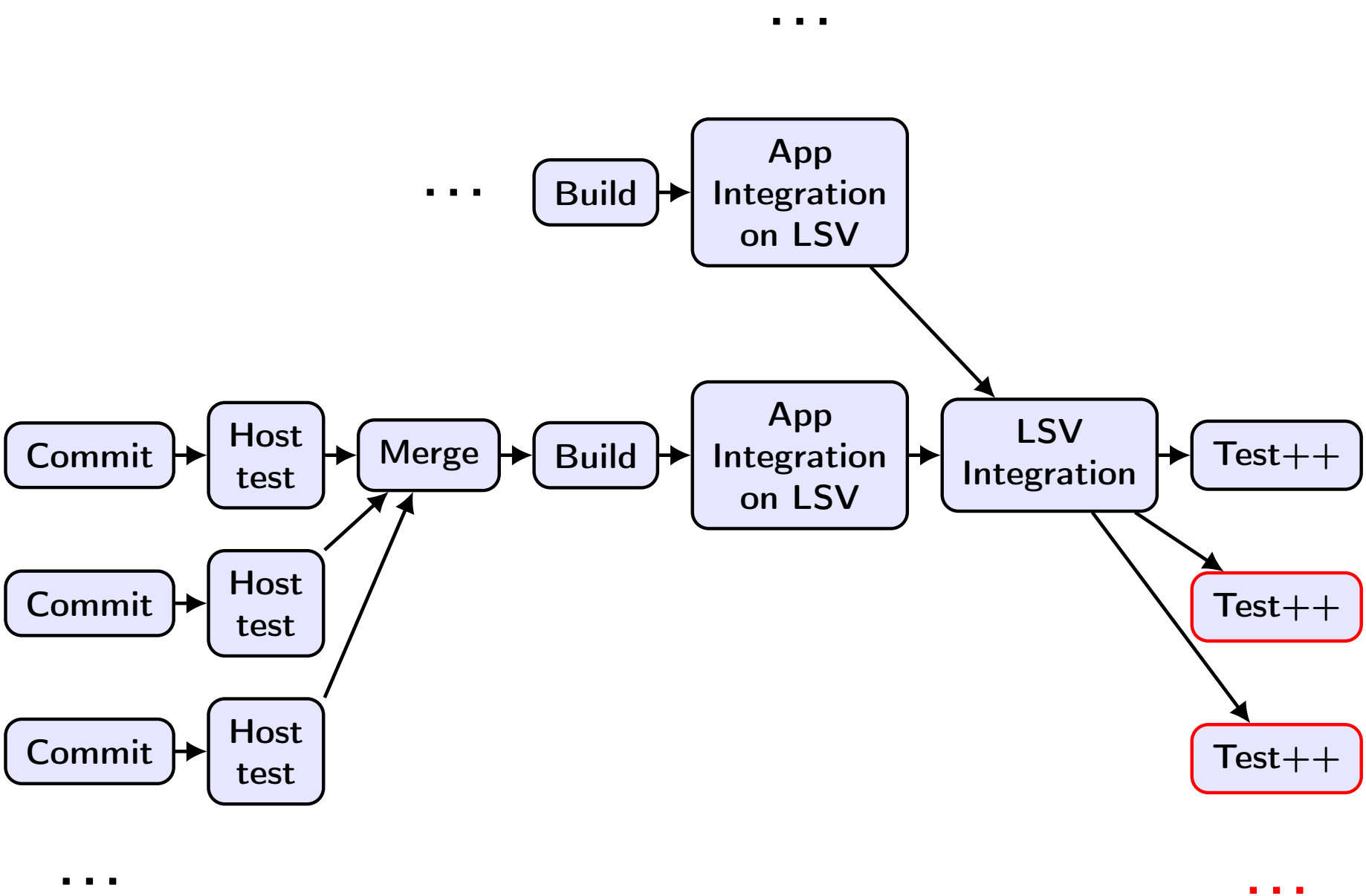


NEXER

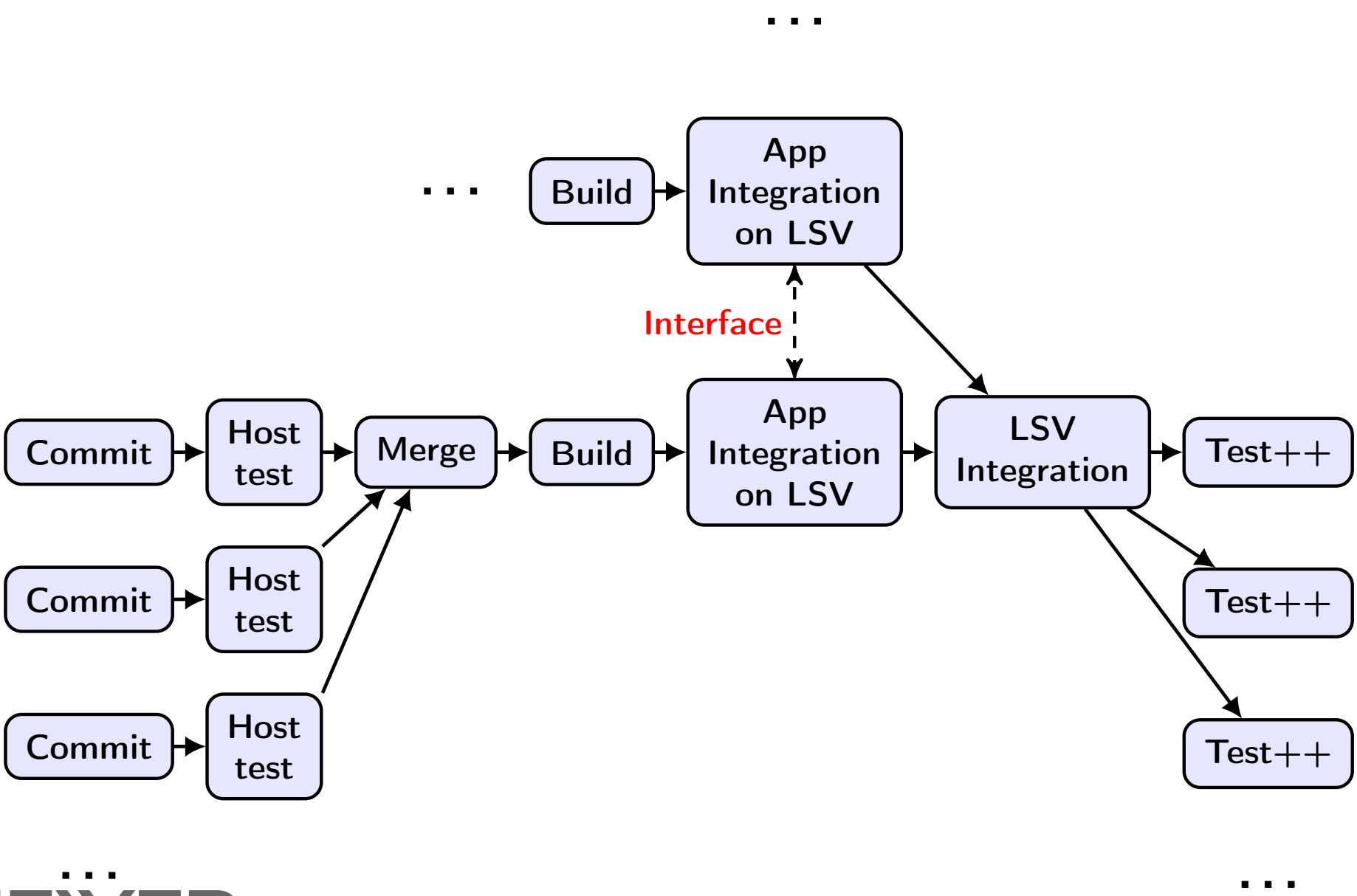
Modularization



Modularization

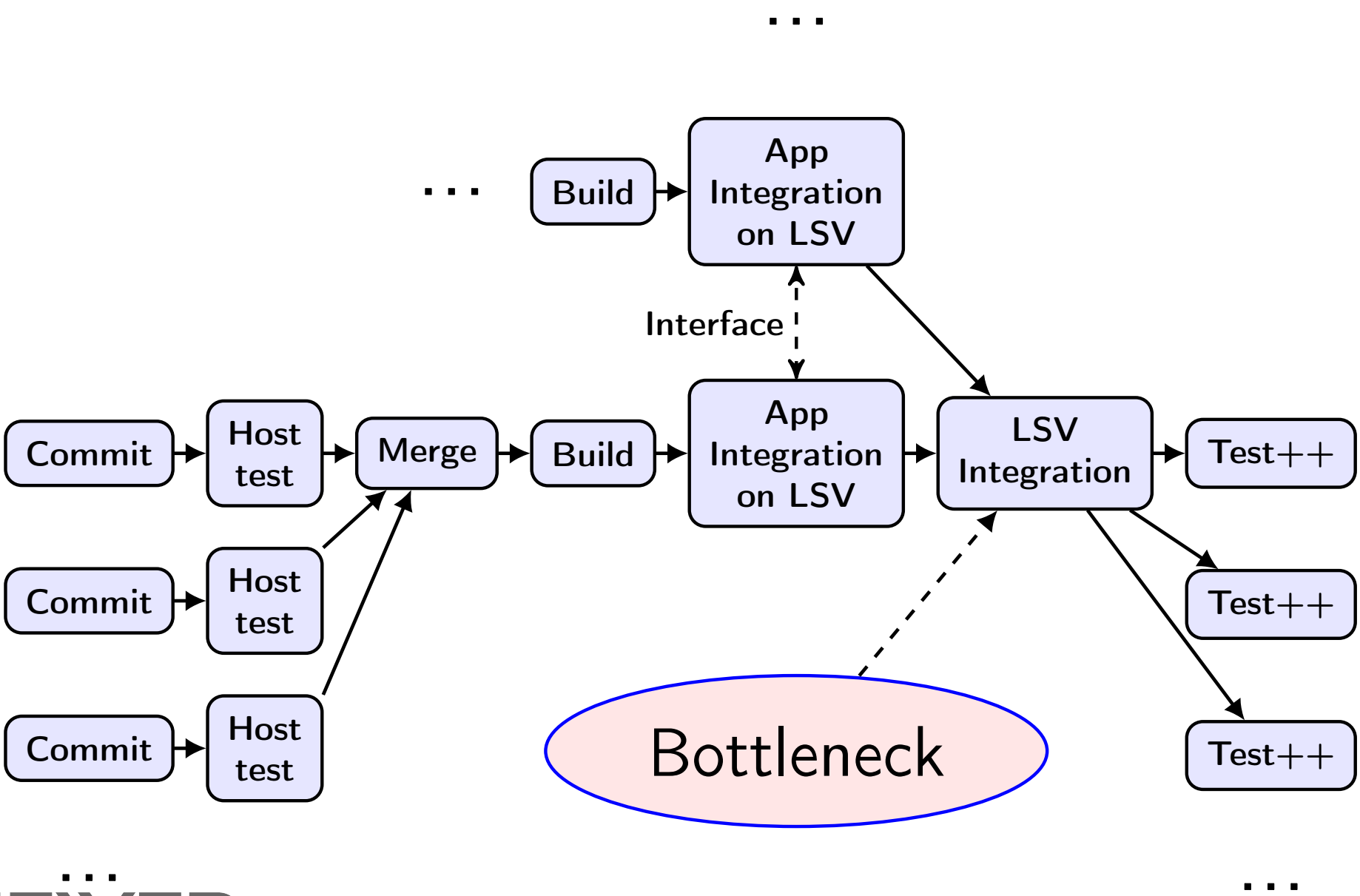


Modularization



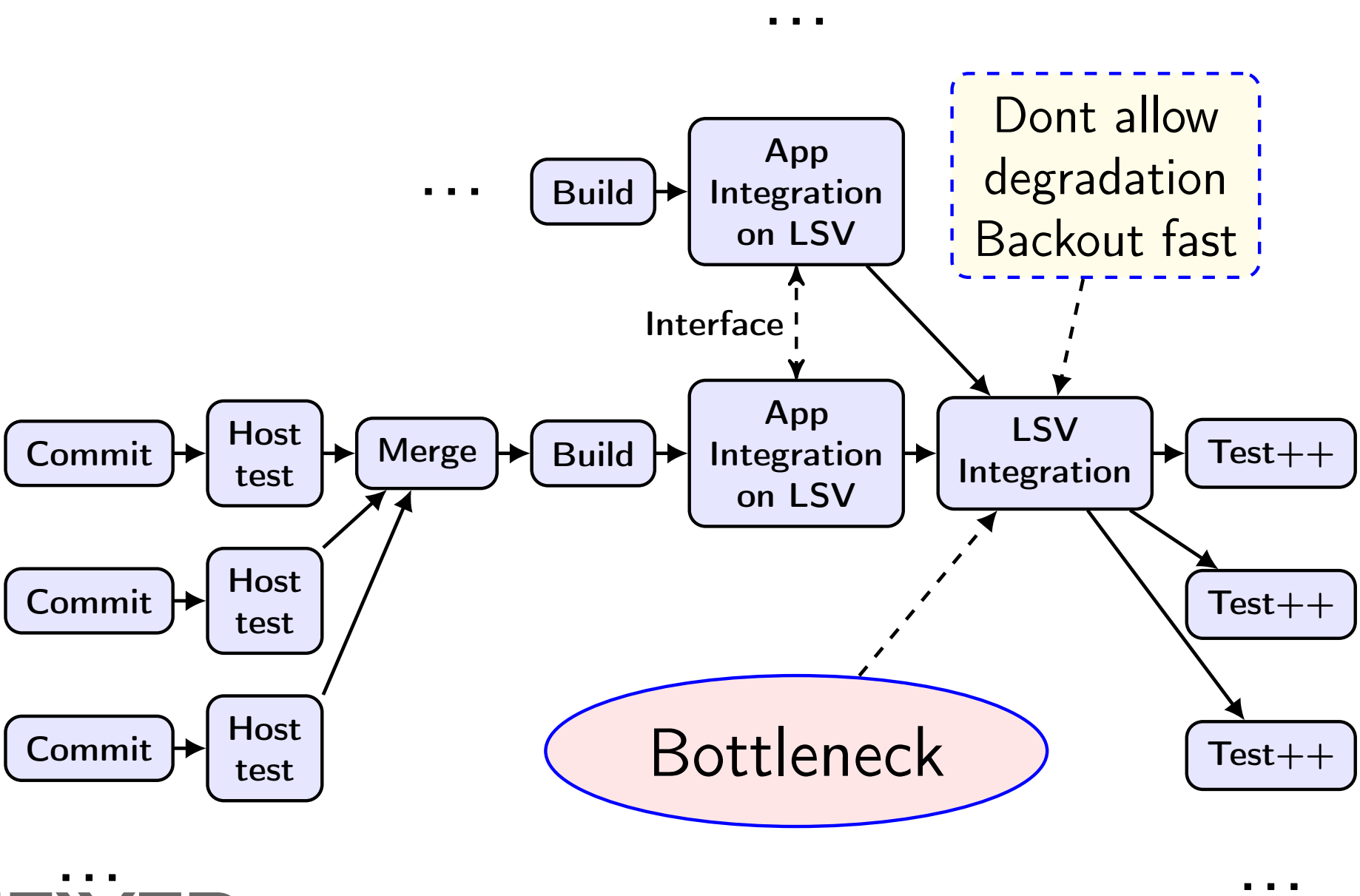
NEXER

Modularization



NEXER

Modularization



Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization thinking

- One developer/app should not stop flow for all
  - Bad quality → You dont get to play
  - Revert/recover first, fix later
- Needed:
  - Clean interfaces
  - Requirements
- (Enabler of Agile!)
  - More defined "sub" responsibilites, better backlogs
  - Sub-orgs solve similar problems → best solution wins!

NEXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

NEXXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

NEXXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

NEXXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

NEXXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

Modularization bad things

NEXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

NEXXER

Modularization bad things

- More spread out
  - Harder to cooperate
  - Multiple solutions to same problem (alignment)
- Permissions
  - "*Why should you have access to my code?*"
- "Box thinking"
  - "My box is perfect" $\rightarrow$ someone elses problem
  - Remember: All working for same goal

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

Way towards Big CI, early actions

1. Parellelism
   - Enables running many tests
2. Build avoidance / caching
   - Don't rebuild source/objects that have not changed
   - Cache objects/build dependencies between consecutive runs
3. Smart testing
   - Many tests $\longrightarrow$ running all cripples CI
   - Only run tests that are related to change
4. Invest in Application and CI architecture
   - Design for testability
     - Divide application into sub responsibilities (modularization)
     - Communicate with backwards compatible interfaces
     - Separation of concerns!

NEXER

Running all the tests

```
$ cd project-x
$ . ci/setup.sh
$ time apps/app00/test/test.sh
## Running tests for /home/solarus/projects/project-x/apps/app00
# Doing complicated arithmetic (aka sleeping) for 8 seconds ...
# Done!
real  0m8.014s
```

```
$ time find -name test.sh -exec {} \;
## Running tests for /home/solarus/projects/project-x/apps/app04
# Doing complicated arithmetic (aka sleeping) for 0 seconds ...
# Done!
...
## Running tests for /home/solarus/projects/project-x/apps/app03
# Doing complicated arithmetic (aka sleeping) for 28 seconds ...
# Done!
real  11m13.586s
```

NEXER

Running all the tests

```
$ cd project-x
$ . ci/setup.sh
$ time apps/app00/test/test.sh
## Running tests for /home/solarus/projects/project-x/apps/app00
# Doing complicated arithmetic (aka sleeping) for 8 seconds ...
# Done!
real  0m8.014s
```

```
$ time find -name test.sh -exec {} \;
## Running tests for /home/solarus/projects/project-x/apps/app04
# Doing complicated arithmetic (aka sleeping) for 0 seconds ...
# Done!
...
## Running tests for /home/solarus/projects/project-x/apps/app03
# Doing complicated arithmetic (aka sleeping) for 28 seconds ...
# Done!
real  11m13.586s
```

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Running all tests

- In this case 50 suites
  - Around 15 seconds to finish $\longrightarrow$ on average 12.5 minutes running sequentially
- Example from one repository:
  - 1 929 test suites
  - (1 035 437 lines of test code)
- Around 15 seconds to finish $\longrightarrow$ about 482 minutes of sequential run time
  - I.e. a work day...

NEXER

Questions?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging

- Remember Big CI Problems:
  - Many tests+developers+apps/Tracking/Intermittency...
- Test failed in App Integration $\rightarrow$
  - Test failed before? (same way!)
    - In same App/other apps?
    - On certain configurations?
  - Intermittent?
    - More intermittent today than last week?

NEXER

Logging cont

- Without data, we are blind to degradations
- Solution: automatic result tracking!
  - Test failure messages, configurations, target log analysis

NEXXER

Logging cont

- Without data, we are blind to degradations
- Solution: automatic result tracking!
  - Test failure messages, configurations, target log analysis

NEXXER

Logging cont

- Without data, we are blind to degradations
- Solution: automatic result tracking!
  - Test failure messages, configurations, target log analysis

NEXXER