



Palantir - a case study
Webworks Roundup 2009

The title of my presentation is a bit outdated “1001 Nightly Builds.” As of this morning’s build report, Palantir has we have 4 main code trunk builds. The help release build, which goes out with our product, has over 1400 successful runs in this branch. 1436 to be exact.

In this simple statement, you can see some of the power of a build infrastructure. Our build infrastructure is integrated with the product, it is a repeatable structure, and it generates data that can be reported and analyzed.

The Presentation

- This is a case study.
- What will this presentation do?
 - ENVIRONMENT
 - REQUIREMENTS
 - IMPLEMENTATION
 - EPUBLISHER
 - SUMMARY
- What will this presentation not do?

Palantir - a case study
Webworks Roundup 2009

A case study of how a documentation build infrastructure that uses ePublisher Software was developed in a small software company.

What it will do:

Explain the people, hardware, and software that make up a documentation build infrastructure

Walk through the build infrastructure at one firm

Describe the role ePublisher Software played in the build infrastructure at one firm.

Identify typical milestones and stumbling blocks in the development process and how to deal with them.

Generate ideas for others about how to develop an infrastructure.

What it won't do:

Teach you how to implement ePublisher Automap

Provide detailed explanations of tool configure or scripts

Who is Presenting

- Experience
- My expertise

18+ years experience in the software industry, mostly in startups.

Working with Webworks on and off since 1998. This is the second build infrastructure I've put together.

I am an expert in Palantir's build documentation build system. This is a subset of our larger build infrastructure.

I do not consider myself an expert in Framemaker, Webworks or even in any of the other tools I will talk about. However, I'll do my best to answer any questions that might arise during the presentation.

ENVIRONMENT

- What kind of company & software?
- What types of documentation are you delivering?
- How many writers?
- How frequently is documentation released?

Every company has a documentation build infrastructure.

The environment at the company determines what kind of infrastructure is appropriate

What Kind of Company & Software

- Palo Alto, CA startup named for Tolkien's seeing stone
- Analytical software:
 - Four end-user graphical user interfaces (GUIs)
 - Three admin GUIs
 - 12 plus servers
 - An API

Palantir - a case study
Webworks Roundup 2009

Begun in 2004, 180 employees

We are growing rapidly which figures into the stresses on documentation.

28 Engineers and 5 writers.

We use the agile development process. Approximately 4 major releases every year.

There are approximately 4 iterations per major releases. Documentation is attempting to release interim material with the iterations.

This means we release documentation externally 4 times a year and have 8 internal releases.

What Types of Documentation are You Delivering?

- Multiple audiences
- 19 documents in the main product line
- Single-sourced, Framemaker 8.0
 - Webworks Help 5.0
 - Adobe PDF
- Javadoc

The audience for our documentation includes:

- Analysts using our Workspace and Web User clients
- Administrators using our Configuration Management and
- Integrators using our Dynamic Ontology Manager client (Java Swing client)
- Programmers using our APIs

WebWorks Help 5.0 installed through our product installers launched from Java Swing client (3 projects)

Web-based software (3 projects)

WebWorks Help 5.0 available from our Devzone Portal. This is a searchable build of all of our products. This single project contains 19 books.

PDF files available for download from our support portal

We also have a Javadoc for programmers to reference. This is its own build.

How Many Writers?

- 4 1/2 full time writers
- 1 part-time doc build engineer
- Average 15 years writing
- Average 6 years experience in Framemaker

How Frequently are You Releasing?

- 4 major releases a year
- Each release contains ~ 4 iterations
- Documentation team releases in tandem

Environment Impacts Infrastructure

- What kind of documentation are you delivering?
- How integrated into the product is the help?
- How frequently are you delivering the documentation?
- How many writers producing how many documents?

RELEASING ENGINEERING GOALS

- What is a release engineering?
- What is the basic process flow of releasing software?
- What are the characteristics of a good releasing engineering process?
- What are the components of a release infrastructure?
- How does releasing documentation compare to releasing software?

Palantir - a case study
Webworks Roundup 2009

It is important to understand where the documentation build fits into the software development process. This understanding can help you:

- Make an argument for creating a build structure if you don't have one
- Find key people in your organization who can move your project forward
- Identify the goals of a good documentation build infrastructure

What is a Release Engineering

“ Sub-discipline in software engineering concerned with the compilation, assembly, and delivery of source code into finished products or other software components. An associated term is software release life cycle.”

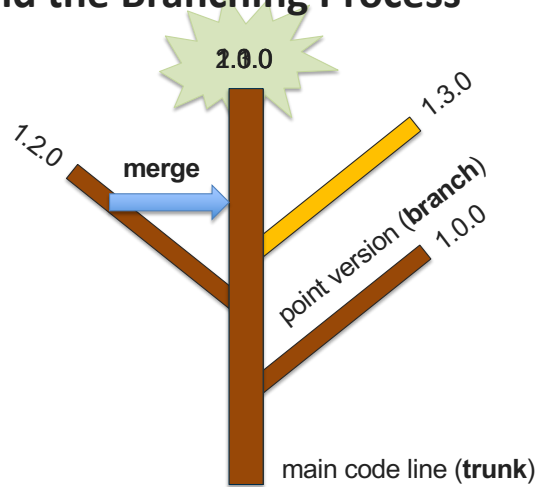
- Wikipedia

Individual engineers “build” all or parts of a product to do unit testing

Release engineers build a version of a product for a release to customers

Typically a team in a company

Understand the Branching Process



Palantir - a case study
Webworks Roundup 2009

Individual engineers “build” all or parts of a product to do unit testing

Release engineers build a version of a product for a release to customers

Typically a team in a company

The Goals of Release Engineering

- Predictable
- Repeatable
- Reliable
- Reportable

Palantir - a case study
Webworks Roundup 2009

How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.

The Goals of Release Engineering

- **Predictable**
 - Repeatabe
 - Reliable
 - Reportable
- *You know what the output of the release will be.*

Palantir - a case study
Webworks Roundup 2009

How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatabe

The process runs in the same manner each time it runs. There is no variation.

Reliable

It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.

The Goals of Release Engineering

- Predictable
- **Repeatable**
- Reliable
- Reportable
- *The process does not vary in execution.*

Palantir - a case study
Webworks Roundup 2009

How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.

The Goals of Release Engineering

- Predictable
 - Repeatable
 - **Reliable**
 - Reportable
- *It succeeds most of the time.*

Palantir - a case study
Webworks Roundup 2009

How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.

The Goals of Release Engineering

- Predictable
 - Repeatable
 - Reliable
 - **Reportable**
- *You collect, store, and review historical data about a release.*

Palantir - a case study
Webworks Roundup 2009

How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

It succeeds most of the time so that other parts of the product can expect

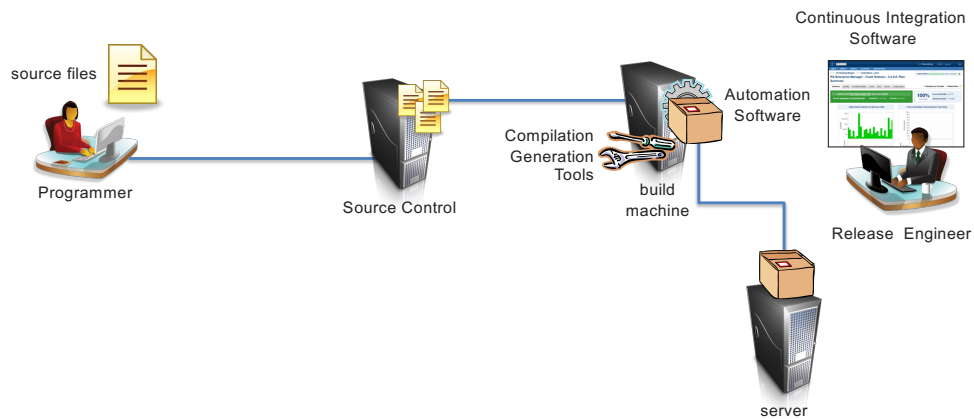
Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.

The Components of Release Infrastructure



Palantir - a case study
Webworks Roundup 2009

You need to understand the human and non-human components required to put together a good release engineering process:

Software release engineer

This is the software person responsible for putting together the software deliverable.

Documentation release engineer

This is the writer responsible for putting together the documentation deliverable.

Source Files

This the raw material that contains your code if you are software engineer. If you are documentation person, this is your Framemaker or other source.

Source change control system

Perforce, Clearcase, SVN

Branches or labels version of the source

Machine for running the build

Almost always a single machine and Windows

Build Automation software

Software that automates the steps for creating one or more documentation deliverables.

Ant, Mayven

Compilation or Generation software or tools

These are the pieces that the build automation calls on to convert the source to a deliverable.

Dependency management software Ivy, Maven

javac, javadoc ePublisher Automap,

Continuous integration software

This is software that takes the output of your build and combines it with the output of others build to make a product.

Bamboo,

How Does Releasing Documentation Compare?

Similarities

- Release dates
- Help is a software component
- Testing
- Versioning

Differences

- Binary source files
- High merging overhead
- Multiple transformation
- Client-side tools
- Windows based
- Documentation domain

Software organizations understand building and managing software releases. In many ways a documentation build looks like a software build. In other ways it does not.

RELEASE ENGINEERING GOALS

- What is a release engineering?
- What is the basic process flow of releasing software?
- What are the characteristics of a good releasing engineering process?
- What are the components of a release infrastructure?
- How does releasing documentation compare to releasing software?

Palantir - a case study
Webworks Roundup 2009

It is important to understand where the documentation build fits into the software development process. This understanding can help you:

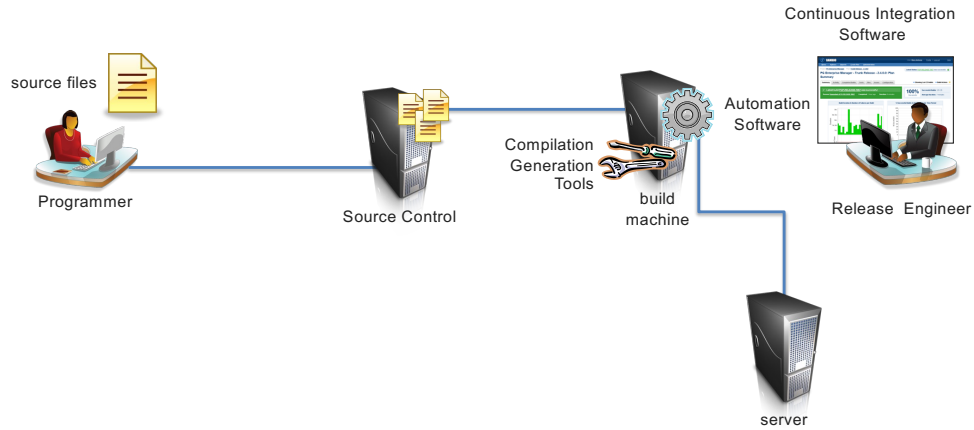
- Make an argument for creating a build structure if you don't have one
- Find key people in your organization who can move your project forward
- Identify the goals of a good documentation build infrastructure

IMPLEMENTATION

This section takes a closer look at Palantir's documentation release engineering implementation.

- the component parts
- the software products used at each stage
- the inputs at each stage
- the out comes at each stage

Starts with Software Release Infrastructure



Palantir - a case study
Webworks Roundup 2009

You need to understand the human and non-human components required to put together a good release engineering process:

Software release engineer

This is the software person responsible for putting together the software deliverable.

Documentation release engineer

This is the writer responsible for putting together the documentation deliverable.

Source Files

This the raw material that contains your code if you are software engineer. If you are documentation person, this is your Framemaker or other source.

Source change control system

Perforce, Clearcase, SVN

Branches or labels version of the source

Machine for running the build

Almost always a single machine and Windows

Build Automation software

Software that automates the steps for creating one or more documentation deliverables.

Ant, Mayven

Compilation or Generation software or tools

These are the pieces that the build automation calls on to convert the source to a deliverable.

Dependency management software Ivy, Maven

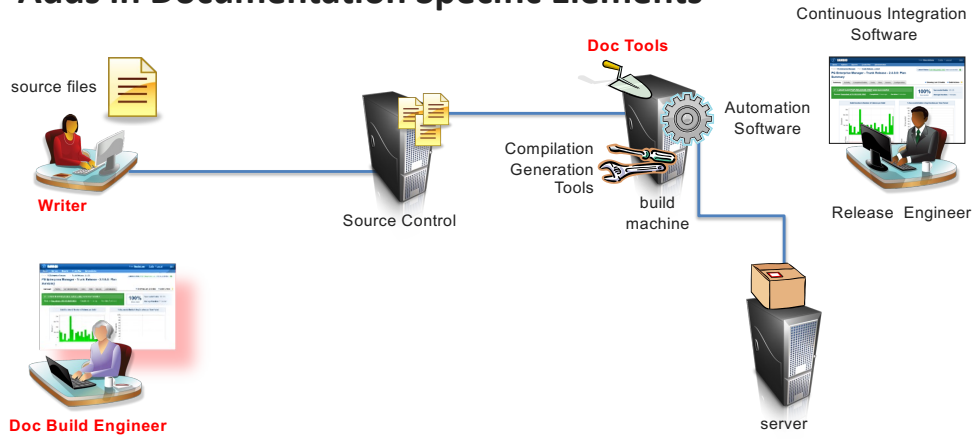
javac, javadoc ePublisher Automap,

Continuous integration software

This is software that takes the output of your build and combines it with the output of others build to make a product.

Bamboo,

Adds in Documentation Specific Elements



Palantir - a case study
Webworks Roundup 2009

Palantir's build structure uses the same tools as the software builds.
Additional tools are layered on that infrastructure.

Designing and Scripting Build Automation

- *Framemaker*
- *Acrobat Professional*
- *Snagit*
- *Visio*
- *ePublisher Pro*
- *ePublisher Express*
- *SVN*
- *Framescript*
- *Ant*
- *Ivy*

Palantir - a case study
Webworks Roundup 2009

1 person with a local machine

Responsibilities

- Designs content template in Framemaker
- Designs help and online formats using ePublisher Pro
- Generates stationery using ePublisher Pro
- Creates ePublisher projects with ePublisher Automap and an editor
- Writes Framescript that performs transformations on content.
- Writes Ant build scripts that automates the build.
- Maintains Ivy files that defines other software in the company dependent on our build.
- Stores the source for safe keeping in SVN.

Designing and Scripting Build Automation

- *Frame*
- *Acroba*
- *Snagit*
- *Visio*
- *ePublis*
- *ePublisher Express*

```
Run eSys.ExtractPathName FullPath(gvActiveFile.Name)
NewVar(gvFolder);

// Loop through all of the files in the folder.
Loop ForEach(File) In(gvFolder) LoopVar(vFile)
// Test for a Framemaker file.
Set vFound = eStr.FindString(vFile, '.backup. ');
If vFound > 0
Delete File(vFile);
EndIf
EndLoop
```

1 person with a local machine

Responsibilities

- Designs content template in Framemaker
- Designs help and online formats using ePublisher Pro
- Generates stationery using ePublisher Pro
- Creates ePublisher projects with ePublisher Automap and an editor
- Writes Framescript that performs transformations on content.
- Writes Ant build scripts that automates the build.
- Maintains Ivy files that defines other software in the company dependent on our build.
- Stores the source for safe keeping in SVN.

Designing and Scripting Build Automation

- *Framemaker*
- *Acrobat Professional*
- *Snagit*
- *Visio*
- *ePublisher Pro*
- *ePublisher Express*
- *SVN*
- *Framescript*
- *Ant*
- *Ivy*

Palantir - a case study
Webworks Roundup 2009

1 person with a local machine

Responsibilities

- Designs content template in Framemaker
- Designs help and online formats using ePublisher Pro
- Generates stationery using ePublisher Pro
- Creates ePublisher projects with ePublisher Automap and an editor
- Writes Framescript that performs transformations on content.
- Writes Ant build scripts that automates the build.
- Maintains Ivy files that defines other software in the company dependent on our build.
- Stores the source for safe keeping in SVN.

Designing and Scripting Build Automation

- *Fram* `<target name="process.individual.help">`
- *Acrok* `<echo message="Build Webworks Help with: ${targetwaj}" />`
- *Snagi* `<echo message="Calling executable." />`
- *Visio* `<exec executable="${automap.shared.location}"`
- `failonerror="true"`
- `timeout="600000">`
- `<arg line="--deployfolder=./build/" />`
- `<arg line="--stagingdir=/temp/" />`
- `<foreach list="${WajSet}"`
- `target="process.individual.help"`
- `param="targetwaj"`
- `inheritall="true">`
- `</foreach>`

1 person with a local machine

Responsibilities

- Designs content template in Framemaker
- Designs help and online formats using ePublisher Pro
- Generates stationery using ePublisher Pro
- Creates ePublisher projects with ePublisher Automap and an editor
- Writes Framescript that performs transformations on content.
- Writes Ant build scripts that automates the build.
- Maintains Ivy files that defines other software in the company dependent on our build.
- Stores the source for safe keeping in SVN.

Authoring

- *Framemaker*
- *Visio*
- *Snagit*
- *Acrobat Professional*
- *SVN*
- *Ixgen*
- *Framescript*

4 writers with local machines

Responsibilities

- Produce content in Framemaker
- Screen captures in Snagit (PNG)
- Graphics in Visio (PNG)

Source Control Software

- Framemaker
- PNG
- Build scripts
- Framescripts
- Master project
- Stationery
- WAJ files



Source Control

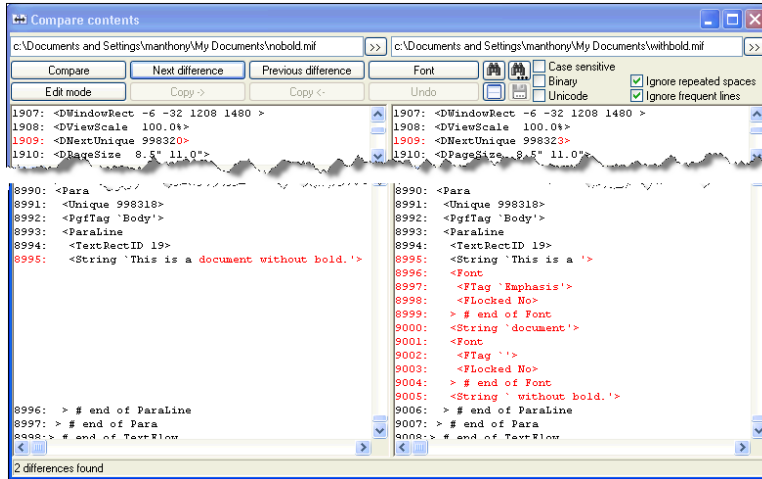
We use SVN this software stores the following sources for the build.

Why aren't we using MIF format so we can merge?

Filenames:

-- keep in mind where your files are stored. Unix file naming conventions. Directory Levels.

Merging Framemaker MIF?



Palantir - a case study
Webworks Roundup 2009

Bamboo -- Continuous Integration Software

- Atlassian product
- Portal into the build system.
- Create a project.
- Define a project plan.
- Execute a plan build.
- Track the build.
- Aggregate and report.

Palantir - a case study
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Bamboo – Project Portal

The screenshot displays the Bamboo Project Portal interface. At the top, there are navigation tabs: Home, Authors, Reports, Create Plan, and Administration. The user is identified as Mary Anthony. The main content area is titled 'Project Gotham Documents: Project Summary'. Below this, a list of build plans is shown, each with a status icon, name, and details. The plans include:

- Fwd PDF - 2.3.4.0 > PGDOC-FWPDF-14**: Ran 22 hours ago, Duration: 60 minutes, Tests: No tests found.
- Fwd Release - 2.3.4.0 > PGDOC-FWDRELEASE-44**: Ran 2 hours ago, Scheduled build, Duration: 6 minutes, Tests: No tests found.
- Stable Devzone - 2.3.3.0 > PGDOC-STABLEDEVZONE-2**: Ran 1 month ago, Manual build by Max Anthony, Duration: 40 minutes, Tests: No tests found.
- Stable Javadoc - 2.3.3.0 > PGDOC-STABLEJAVADOC-7**: Ran 1 month ago, Manual build by Max Anthony, Duration: 10 minutes, Tests: No tests found.
- Stable Release - 2.3.3.0 > PGDOC-STABLERELEASE-9**: Ran 1 month ago, Manual build by Max Anthony, Duration: 60 minutes, Tests: No tests found.
- Test Doc > PGDOC-TESTDOC-1**: Ran 3 days ago, Initial clean build, Duration: 17 minutes, Tests: No tests found.
- Trunk Devzone - 2.4.0.0 > PGDOC-DEVZONE-201**: Ran 10 hours ago, Scheduled build, Duration: 28 minutes, Tests: No tests found.
- Trunk Forward - 2.4.0.0 > PGDOC-FWDOC-32**: Ran 1 week ago, Manual build by Max Anthony, Duration: 36 minutes, Tests: No tests found.
- Trunk Forward PDF 2.4.0.0 > PGDOC-TRFWPDF-4**: Ran 1 week ago, Manual build by Max Anthony, Duration: 50 minutes, Tests: No tests found.
- Trunk Javadoc - 2.4.0.0 > PGDOC-JAVADOC-315**: Ran 57 minutes ago, Dependent of PGDOC-PDF-16, Duration: 12 minutes, Tests: No tests found.
- Trunk PDF - 2.4.0.0 > PGDOC-PDF-16**: Ran 1 hour ago, Dependent of PGDOC-RELEASE-1498, Duration: 21 minutes, Tests: No tests found.
- Trunk Release - 2.4.0.0 > PGDOC-RELEASE-1498**: Ran 1 hour ago, Scheduled build, Duration: 17 minutes, Tests: No tests found.

At the bottom of the screenshot, it says 'Powered by Atlassian: Bamboo version 2.2.2 build 1207 - 12 May 09' and provides links for 'Report a problem', 'Request a feature', 'Contact Atlassian', and 'Contact Administrators'.

Palantir - a case study
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

ENVIRONMENT | REQUIREMENTS | **IMPLEMENTATION** | EPUBLISHER | SUMMARY

Bamboo – Plan Page

The screenshot displays the Bamboo web interface for a specific plan. At the top, a navigation bar shows 'ENVIRONMENT | REQUIREMENTS | **IMPLEMENTATION** | EPUBLISHER | SUMMARY'. Below this, the page title is 'Bamboo – Plan Page'. The main content area shows the plan name 'Project Gotham Documents - Fwd Release - 2.3.4.0: Plan Summary'. A status bar indicates 'Build PGDOC-FWDRELEASE-45 is currently building' with a progress indicator showing 64% successful builds out of 16/25. Below the status bar, there are two charts: 'Build Duration & Number of Failures per Build' and '% Successful Builds & Avg Duration per Time Period'.

Palantir - a case study
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Bamboo – Plan Page

The screenshot displays the Bamboo web interface for a specific build plan. At the top, there are navigation tabs: Summary, Activity, Completed Builds, Tests, Files, Issues, and Configuration. The 'Activity' tab is selected, showing the build's progress. The build is identified as 'PGDOC-FWDRELEASE-45' and is currently in a 'building' state. A progress bar indicates that the build has been running for 41 minutes and has approximately 23 minutes remaining. Below the progress bar, the live activity logs are visible, showing the execution of various pre-build actions such as 'Build Number Stamper', 'VCS Version Collector', and 'Pre Command Agent Runner'. The logs also show the start of the build process with AntBuilder.

Palantir - a case study
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

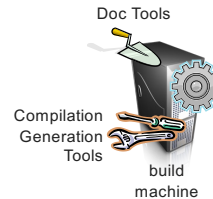
Every plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Build Machine

- Windows machine
- Must meet ePublisher requirements
- Runs a Bamboo agent process



Palantir - a case study
Webworks Roundup 2009

A Windows client machine. This machine:

Runs a Bamboo agent process: waiting to build

The agent process calls the Ant Build script

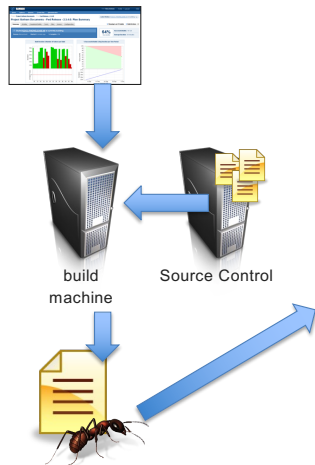
Ant tasks:

- Clean the directories from the previous build
- Setup the ePublisher stationery directories
- Preprocess the Framemaker files with Framescript
- Call ePublisher Automatp to generate the help and online deliverables.
- Call Adobe Acrobat to generate the PDF.
- Save the files to the Ivy repository
- Copy the build results to the test server

The Bamboo agent tracks the build and collects the log

The Bamboo agent collects and provides the artifacts from the build

The Build Process



1. Sets global properties such as the libraries to use and version numbers.
2. Reads in a properties file.
3. Cleans the build directory.
4. Creates a set of temporary directory and unzips **ePublisher** stationery into it.
5. Creates a distribution directory.
6. Preprocesses the files setting conditional tags with a Framescript.
7. Calls **ePublisher** Automap from the command line to process the files.
8. JARs the **ePublisher** distribution for each help set.
9. Copies the JARs to an IVY server.
10. Copies the distributions to a test server.

Palantir - a case study
Webworks Roundup 2009

A Windows client machine. This machine:

Runs a Bamboo agent process: waiting to build

The agent process calls the Ant Build script

Ant tasks:

- Clean the directories from the previous build
- Setup the ePublisher stationery directories
- Preprocess the Framemaker files with Framescript
- Call ePublisher Automap to generate the help and online deliverables.
- Call Adobe Acrobat to generate the PDF.
- Save the files to the Ivy repository
- Copy the build results to the test server

The Bamboo agent tracks the build and collects the log

The Bamboo agent collects and provides the artifacts from the build

Remaining Components

- IVY Dependency Manager
 - Contains the output of all builds
 - Different builds in the system pull from this
- Staging Server
 - Web server on the network
 - Updated with each build
- Wiki

IVY Dependency Manager

Receives the output of the build and serves it to other builds that require that output.

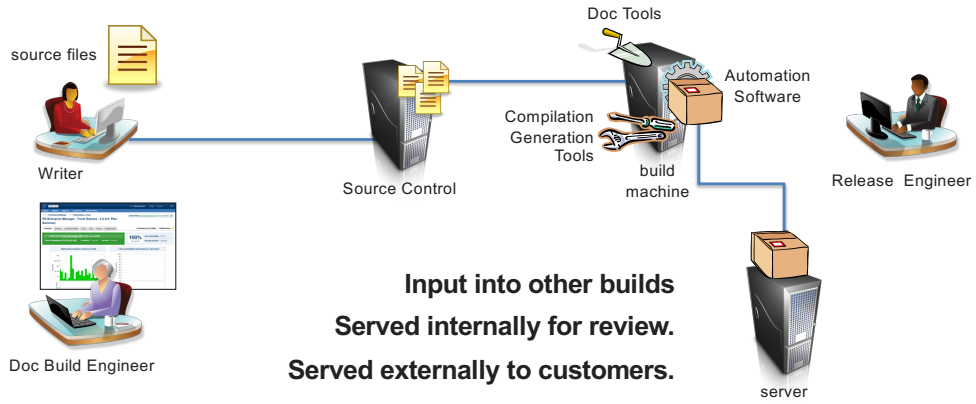
Staging Server

The Ant build copies the results of the build across the network to a web server.

Wiki

Provides links to the material on the web server.

Summary



Palantir's build structure uses the same tools as the software builds. Additional tools are layered on that infrastructure.

EPUBLISHER

This section takes a closer look at each of the role of the ePublisher Software in Palantir's build infrastructure.

ePublisher Pro

- Supports two output formats WW Help 5.0 and Eclipse
- Used to create two stationery formats with and without Google Analytics
- A zip of the Master Project is under source control.
- A zip of the Stationery is under source control

Palantir - a case study
Webworks Roundup 2009

The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt

ePublisher Master Project

- Zip file is in source control
 - Files
 - Formats
 - Targets
 - genericMasterProject.wep
 - README_FIRST.txt
- Unzip to local drive to update.

Palantir - a case study
Webworks Roundup 2009

The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt

ePublisher Stationery

- Zip file is in source control
- Unzipped during build
 - Files
 - Formats
 - Targets
 - genericStationery.wxsp
- Ant unzips during build

Palantir - a case study
Webworks Roundup 2009

The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt

ePublisher Express

- Support for Automap
- Individual writers do not use

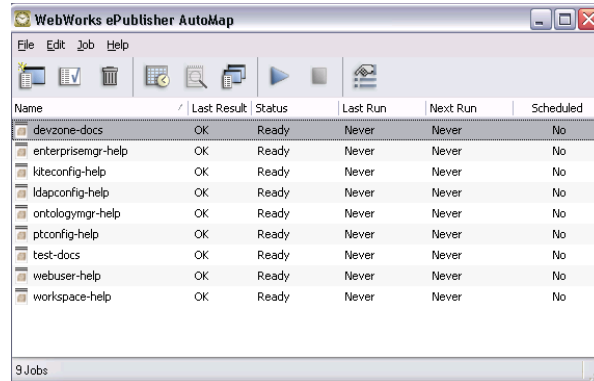
Is used to support the Automapper. Our individual writers do not use this product.

Reasons for this decision:

- Reduces the number of tools writers need to learn. Too onerous given our release cycle
- Writers should focus on learning and writing about Palantir products.
- Writers are all located locally with easy access to the build machine.
- Writers review output from the frequent builds.

ePublisher Automap

- Registers projects
- Launched from the command line via Ant



Palantir - a case study
Webworks Roundup 2009

Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.

Automap Projects, Creation, and Express

- Express Project
 - Requires opening express and creating a project.
 - Select targets
 - Schedule
 - Cannot change document set.
- Stationery
 - Specify stationery
 - Identify documents
 - Select targets
 - Schedule
 - Can change document set by hand edits.

Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.

Webworks Automap Job (WAJ) Files

- Form the basis for each project
- The <Files> tag defines the documents.
- <MergeSettings> defines the grouping for the navigation
- Specify a context id.

Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.

ePublisher at Palantir

- Zipping Pro projects and stationery instead of source control of folder hierarchies
- Emphasis on stationery-based Automap over Express projects
- Reduce writer over head
- Called from a command line rather than Automap schedules

Palantir - a case study
Webworks Roundup 2009

Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.

SUMMARY

- This section discusses:
 - Areas we are seeking to improve
 - Pros and Cons

For the Future

- Build Frequency
- Testing
- Reporting

Build Frequency

Want to provide a facility for individual writers to launch a personal build through our infrastructure. This will enable tighter build/review cycles near the end of a build iteration.

Reporting

Are not currently using the Reporting capability of ePublisher. This is a powerful tool we want to make use of. Time limitations not in terms of implementing but in terms of taking the results and managing them.

Testing

Writing scripts that test builds before they are submitted.

Pros of a Solid Doc Release Infrastructure

- Similar processes engender more knowledgeable working relationships between the teams.
- Automation means progress is reportable and measurable
- Documentation incorporated in the product so engineers build doc into their code right from the start

Palantir - a case study
Webworks Roundup 2009

For example, when a new release is anticipated, because we are “in the loop” with the release engineering team, we have a faster track as writers to hearing about it.

By knowing how the release engineers work, we can foster improvements in their processes and speed the delivery of our documentation process.

Engineers learn what goes into producing the documentation.

Pros of a Solid Doc Release Infrastructure

- Nightly build means instant feedback
- Matching development process timing in an agile cycle

Palantir - a case study
Webworks Roundup 2009

For example, when a new release is anticipated, because we are “in the loop” with the release engineering team, we have a faster track as writers to hearing about it.

By knowing how the release engineers work, we can foster improvements in their processes and speed the delivery of our documentation process.

Engineers learn what goes into producing the documentation.

Challenges from the Experience

- Organizational support for the time/personnel required to build and maintain the system
- Getting buy-in from the writing team.
- Inability to merge. We can mimic but not quite.
- Client software with automated build.
- Windows restriction.

Documentation builds take knowledge about documentation products, processes, and scheduling. It takes a person who is both a writer and a bit of an engineer.

Writing team can be resistant to the processes automation. Requires new tools and new ways of doing things.

Positive Process Impacts

- Nightly build means instant feedback
- Matching development process timing in an agile cycle

Acknowledgements

- Richard Neale, Palantir Release Engineer
“Thank you Obi-wan”
- Vicky Bialas & Liz Keene, Arcsight Doc Build Team
“Thank you both for pithy comments in reviewing my presentation. And for sharing your own experience and knowledge.”

QUESTIONS