Software Development In the Cloud

Cloud management and ALM

First published in Dr. Dobb's Journal, February 2009: http://www.ddj.com/development-tools/212900736

Nick Gulrajani is a Senior Solutions Architect with CollabNet and can be contacted at nicholasg@collab.net
Darryl Bowler is a Senior Systems Architect Consultant with CollabNet and can be contacted at dbowler@collab.net

The latest software development methodologies such as Agile require a more collaborative and dynamic environment for teams to work. Short iterations and continuous integration mean results and feedback are shared continuously, system configurations change often, and working over time-zones is common. To achieve this flexibility, more adaptable computing resources are needed. Nick and Darryl provide an example use case and tools you can use in your environment, bringing together the necessary ingredients for software development in virtual private clouds, or even public clouds like Amazon EC2.

How does cloud computing change software development?

Similar to how virtualization began to take hold in the engineering lab, cloud computing is taking root with the more experimental crowd in software development. The reason for this is obvious: development teams are quick to jump on to any leading edge technology that solves their challenges.

New methodologies require more flexibility

These new challenges have come to the forefront because software development methodologies have evolved to require more dynamic, flexible tools and processes. This in turn means the development computing environment requires more adaptability. Systems specifically need to accommodate shorter project sprints, be less static and more configurable-on-the-fly, and support collaborative principles. In short, on-demand resources that can be shared across teams, managed by development, and have traceability across projects.

Access to flexible, on-demand cloud computing resources, either from a virtual private cloud within the corporate data center or from public clouds, can provide such a flexible environment. While clouds are interesting on their own (who doesn't want all of Amazon's computing resources available to them?), having tools for managing the cloud resources *specifically* for development tooling and workflow is key. *Cloud management for development* brings teams this level of control and visibility, necessary in the new landscape.

Cloud management for development

With cloud management for development, teams ultimately drive the allocation and provisioning of their systems, on-demand, as they need them. They can utilize the pooled physical and virtual machine capacity of a cloud for more flexible automation and reuse across projects. Which means less time spent configuring and finding errors when software moves from one stage to the next.

The benefits of thinking of development resources in this way – servers as 'clouds' or 'pools' of virtual resources and version controlled configurations – is several fold for development teams:

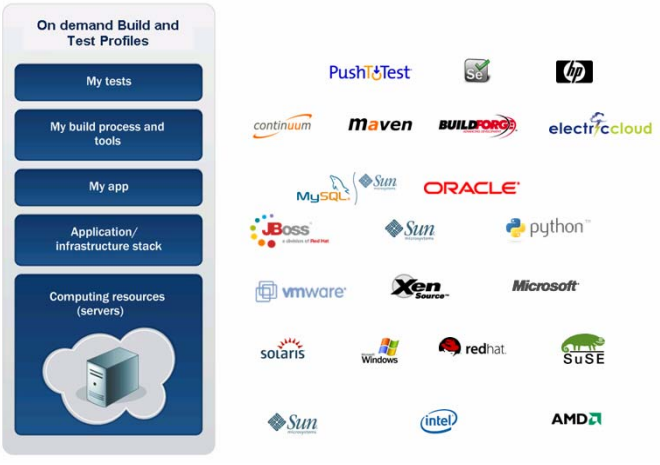
- Managing and controlling entire development, build, and test process from one interface means capacity is able to be monitored and optimized.
- The profiles (configurations) are stored and managed as reusable assets across a project or across multiple projects, easily accessible by any developer in the project, or in the company.
- Visibility from the project team level and the management team as to how the resources are being used, with the ability to charge back resources per project if desired.

Cloud management and virtual private clouds: in your data center and beyond

For the purpose of this article, the focus is on how to manage *virtual private clouds* for Application Lifecycle Management (ALM). We generally define virtual private clouds as groups of public or private server pools from your corporate data center or from public clouds like Amazon EC2. To these server pools, a developer would apply some software stack or configuration required for their task at hand. A typical enterprise use case for *managing their cloud* might be:

1. Engineering or Project Managers set up what clouds a project may have access to, to control costs, manage security, supplement resources during peak use.
2. Individual hardware resources can be assigned to a 'cloud' that now has the capacity of their combined resources. Dell class servers could be allocated to a development cloud, while HP blades with a higher service level agreement and security could be allocated to a production cloud. The cost associated with a machine in the former would be $.10 per hour while the latter, $.25 per hour.
3. A project manager (or admin) allocates portions of either cloud to projects as the development or production configuration. He can limit projects to certain clouds; i.e. for development purposes only services from development cloud can be used.
4. Finally, cost accounting data can be derived by tracking usage by project and user, so managers can optimize which assets are being used for what, at varying stages, and projects.
5. Amazon EC2 can be used to extend resources temporarily (and at a very low cost).

Figure 1: Build and Test Profiles



Introducing *Development Services*

To complete the concept of cloud management for development, let's also introduce the concept of *Development Services* or *Build and Test Services*.

Development Services consist of code, build and test tools, applications, and infrastructure stacks that can be stored and managed as configurations or profiles, and applied to an available server. These profiles can be accessed and used globally and version controlled for consistency across the application development lifecycle. So *Development Services* are simply: *Software configurations or profiles applied to an available server in the cloud, on demand.*

Development in the cloud: A practical use case

One use case for software development in the cloud combines the agile best practice of *continuous integration* (CI) and some common collaborative development tooling that has been used across companies of varying sizes, locations, and industries. Meaning this is quite scalable and flexible around the size of your development team and type of products you are building.

Continuous Integration

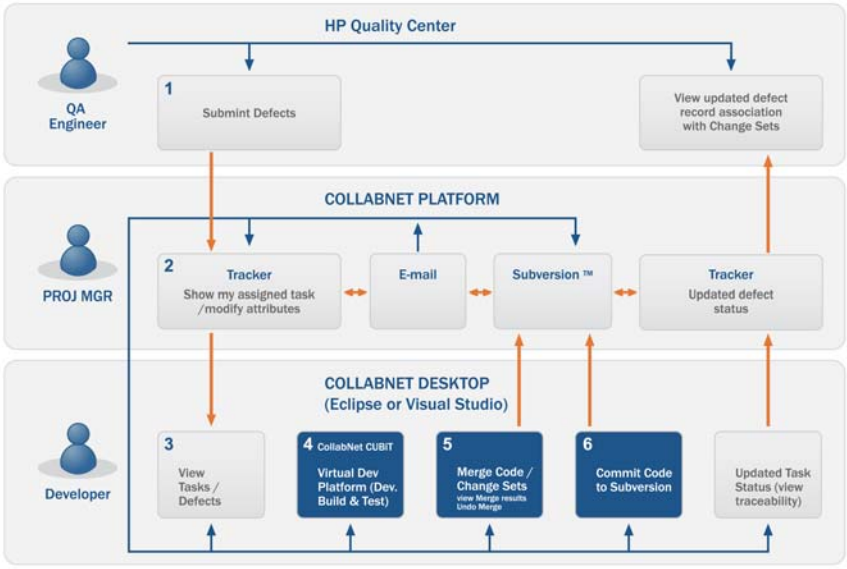
Continuous Integration is a key concept to agile practices. It is based on the philosophy of *why wait until the end of a project to see if all pieces of the system will work*? Every few hours the system should be fully integrated, built and tested with all the latest changes so adjustments can be made. This is instead of waiting until modifications pile up or data is replicated to the integration site from various locations or individuals.

The cloud development use case encompasses the flow of defects/requirements through phases of development/builds/tests and back to submission of new requirements or defects by various stake holders. Automation at any point possible is a key capability, including the ability to 'turn on' and 'rip down' virtual or physical systems as needed, in a cloud.

The workflow

Figure 2 illustrates the workflow for development in the cloud through the perspective of the various contributors, along with their collaborative and cloud management tools:

Figure 2: Continuous Integration in a virtual private cloud



1. Business Analyst /Quality Engineer (BA/QE) submits defects/requirements.
2. Project Manager (PM) picks tasks, sets priorities and assigns it to development.
3. Developer opens his favorite IDE and views his task. Begins to work on the defect, writes code.
4. May use the cloud management platform to build and test his code.

5. Merges code and change sets.
6. Commits his code to a source code management tool (here, Subversion). This triggers a Continuous Integration (CI) that takes place using the cloud management and build automation tool (Figure 3).
7. CI tool monitors for code changes (See Figure 3).
8. Upon build failure, defect tracker is updated and notification is sent to the Development Team (Figure 3).
9. Upon successful build, the defect tracker is updated automatically. If the test succeeds test results are e-mailed to PM/QE. If the test fails QE is notified (Figure 3).

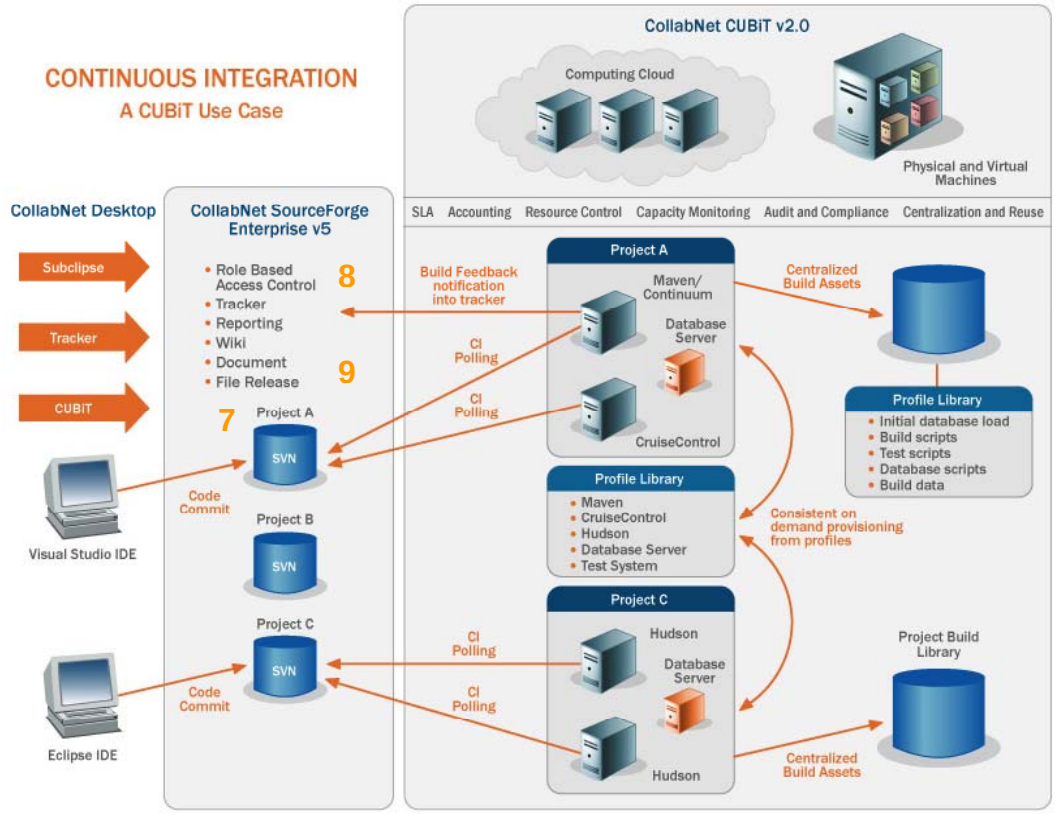
Virtual private cloud configuration

So what part of this happens in the virtual private cloud?

- Step 4 is where the developer may configure his own cloud system on demand for build and test.
- Steps 7, 8, and 9 - the CI tool and build automation - runs on the cloud systems.
- Once a build is successful, the artifacts are uploaded automatically to the Project Build Library on the cloud systems. The Project Build Library (PBL) stores files created and used by the CI process, and can be shared with others who may need to access the build results.
- A test system can be dynamically provisioned and build artifacts are downloaded and tested automatically.

Figure 3 illustrates what the CI in your private cloud might look like, with virtual and physical machines and tools for source code, tracking, build, and test. The following section will explain how to build your own.

Figure 3: Software Development in the cloud workflow and stakeholders



Setting up your cloud development environment

Now how to do it: This section overviews some basic tools and principles to consider when modifying your own environment for development in the cloud.

Tools and Framework

First, some basic tools and principals should be considered.

Basic team processes

- All stake holders should be able to see and make progress on the code and executables in real time without having to replicate data or have to change ownership
- A version control system should be integrated with build automation tools
- Simple branching strategy (parallel development) for development, integration and release work
- Make incremental changes and test often
- Focus on small releases that provide most business value
- Utilize aggressive milestone management, such as the agile imperative of *time boxing*.

Time boxing, short iterations, & cloud management

Time boxing deserves a few more sentences as the technique is common in agile. It means splitting up the project into a number of separate time periods, each 2-6 weeks long. Each of the separate parts has their own deadline and budget; where deadlines are fixed but deliverables are adjusted. To meet the fast turnaround times and splitting of the project, engineering has to be able to allocate and configure systems quickly, when needed. If constrained by access to these systems or incorrect configurations, the short iterations are impacted. *On-demand cloud management server pools can mitigate this*.

Tools

Taking advantage of software development in the cloud does not require massive retooling – that's one of the benefits. You may already be using many of the basic collaboration and change management tools that are commonly deployed by software teams world wide. (As a disclaimer, many of the tools we talk about are those that CollabNet sells.) *Table1* lists out the tools.

Table 1: Tools for Software Development in the Cloud

Tool	Activity	Example
Project Activity Tracking/ Collaboration	An integrated suite of Web-based issue tracking, project management, and collaboration tools such as wikis and document management.	CollabNet SourceForge
Automated build tools	Build management and acceleration tools	CruiseControl, ANT, Hudson and Maven
Automated test tools	Defect submission, automated tracking, etc.	HP Quality Center, PushToTest
Source Code Management	Commit Code, Branch and Merge	Subversion
Integrated development environment (IDE)	Developer desktop tool that interfaces with source code management, debug, unit testing and other tools	Eclipse, Visual Studio
Feedback mechanism	Notification system for status of builds, tests, issues, etc. across the lifecycle. Email and SMS are good, however an integrated tracker system enables more auditability of changes, can use knowledge threading, and can associate build defects with source code and/or other artifacts.	CollabNet SourceForge Tracker, Email, SMS, wikis
Virtual private cloud management	Physical and virtual machines that can be flexibly managed by software development teams. Profiles (configurations) can be version controlled and managed.	CollabNet CUBiT
Project Build Library (PBL)	Centralized area for daily builds, as CI is not always sufficient for comprehensive testing. In addition, daily or weekly build, integration, and test on a clean 'production-like' system, configured with a standard production profile. PBL helps to automate this task.	CollabNet CUBiT

Requirements for the tools in general include:

- Support for heterogeneous environments: so teams can have flexibility across their organization, based on their specific project requirements
- Adaptive to current workflows and tools: Developers, Build, QA engineers want to stay in their environment they are working
- Access to complementary tools and configurations across the development lifecycle

Development services and cloud management

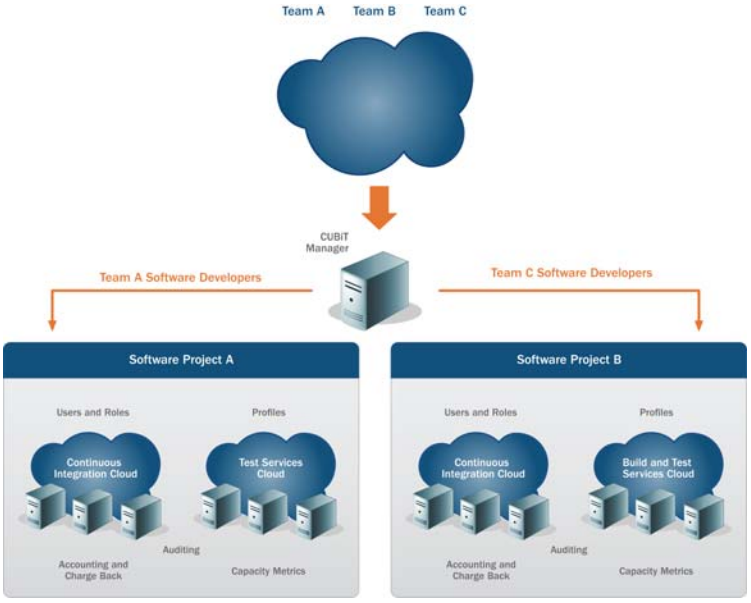
CollabNet CUBiT- CollabNet's *Cloud Management for Development Services* - can be configured to manage a virtual private cloud with your own corporate computing resources, expanding to a public cloud such as Amazon EC2 if desired. This type of capability is a key component for automating any development environment. With such automation, the overhead for software teams to obtain and manage computing resources is greatly reduced with the *Development Build and Test Services* described earlier. New systems can be provisioned from the pre-defined profiles within minutes while maintaining corporate security, auditability, and traceability.

Design goals for the cloud environment are:

- Each team must have autonomy over their own resources, and must have minimal delay in provisioning their own systems
- Strict access controls so that resources are dedicated to project teams or to dedicated purpose (such as build or test)
- Charge back for resources used. Even if the organization doesn't charge back, teams can begin to understand the resource costs associate with their projects
- Using agile processes such as continuous integration (CI)
- Automated build and test

This diagram overviews how CollabNet CUBiT manages development clouds of build and test services – which are a combination of physical and virtual machines with profiles. Software developers within their own projects have access to their own dedicated systems that are dynamically shareable amongst projects members. CUBiT management features includes visibility of all system resources within cloud, auditing, capacity monitoring, role based access and accounting with charge back. Profile management enables version control and traceability, for easily configuring pre-defined development stacks within minutes.

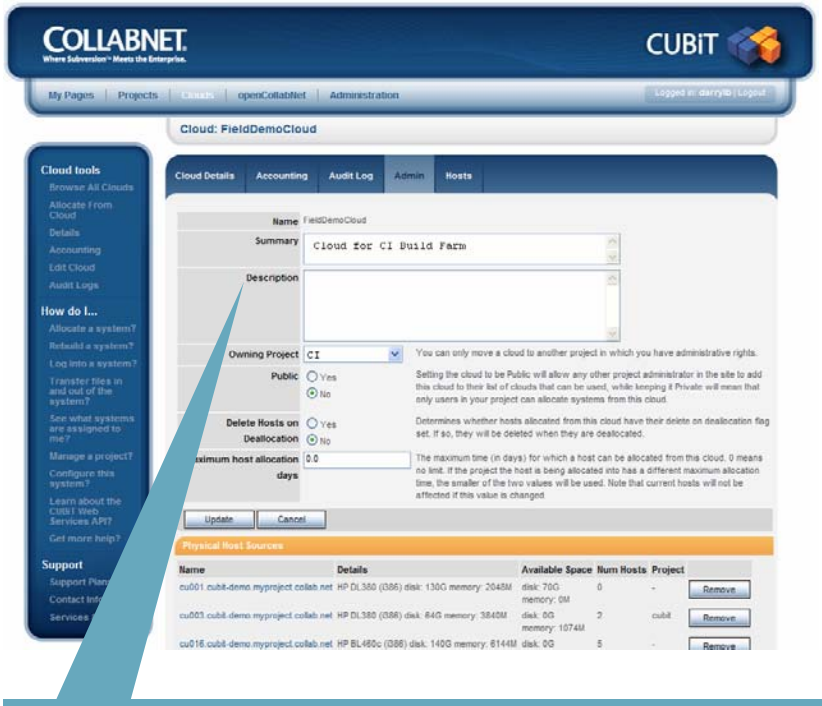
Figure 4: Cloud Management with CollabNet CUBiT



Step 1: Configuring a virtual private cloud

Using CollabNet CUBiT you can configure multiple clouds for different use cases, and have the ability to control the usage of those clouds. For example we could create clouds for continuous integration build systems and assign that as an exclusive resource to that project; or share it among any number of projects.

Figure 5: Configuring a virtual private cloud



CUBiT Manager Web UI Lets You Easily Add a New Cloud

Name – Name of cloud, e.g. "CI Build Farm"
Summary
Description
Owning project – This is important as we will be defining a dedicated build farm cloud, that will only used by developers in a dedicated project.
Public – Specifying "NO" will determine that the cloud is private
Delete Host on Deallocation - When systems use time expires they will automatically delete
Maximum host allocation – The maximum use time of an allocated system
Physical sources – Physical machines that will become the cloud

Step 2: Giving developers easy access to cloud resources through their IDE's

CollabNet provides a free plug-in to popular IDE's such as Eclipse and Visual Studio. This allows developers to seamlessly access the source code management tool, collaboration platform, and development services (CollabNet Subversion, SourceForge Enterprise, and CUBiT respectively) without having to leave their IDE.

Figures 6 and 7 show that from the IDE, you can browse all system resources within a cloud or CUBiT domain, physical or virtual systems. The management interface can be viewed directly from within the IDE; as well the systems can be securely accessed.

Figure 6: Browse and mange from your IDE

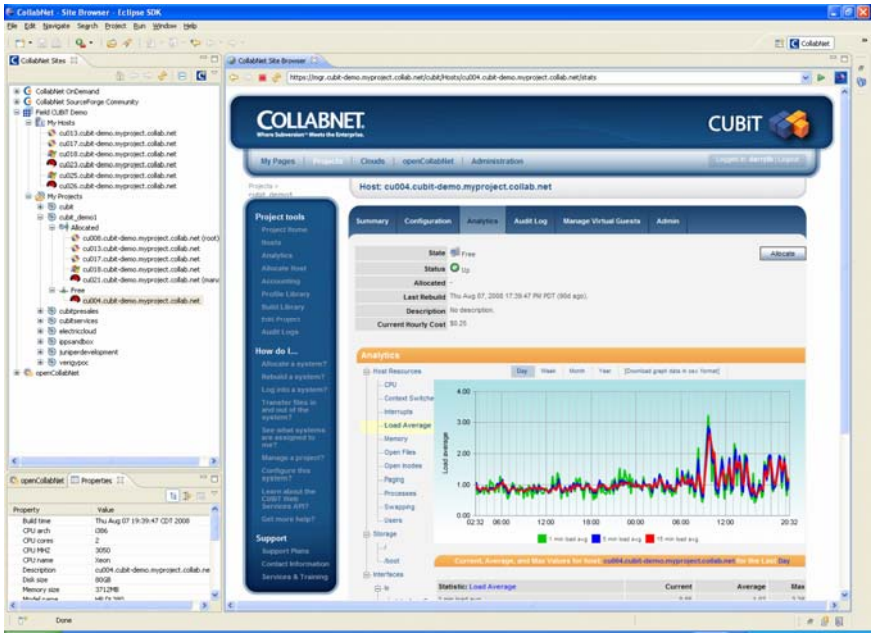
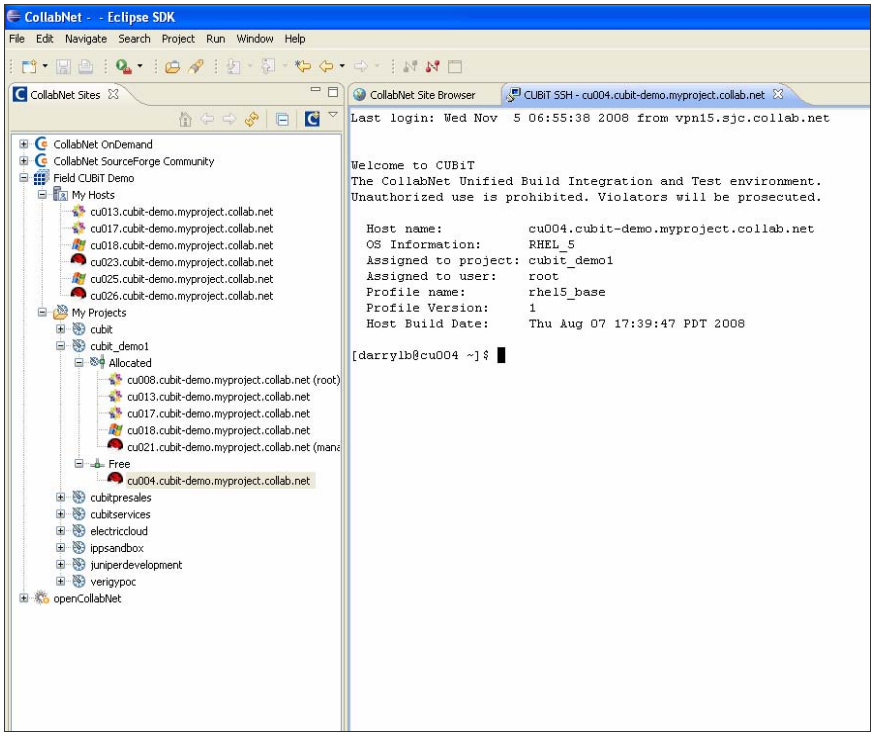


Figure 7: Remote access



Step 3: Creating reusable profiles for on demand build and test services

By creating reusable pre-defined configurations explained as profiles earlier, systems can be consistently provisioned on demand, hence the concept of build and test services. Profiles are defined in an XML format and are maintained in a Subversion repository under version control, so systems can be restored exactly to a previous state if needed.

This profile snippet demonstrates how a continuous integration build system can be provisioned from CUBiT. This example will install Java SDK, CruiseControl, Apache Beehive, Apache Derby and Tomcat, a full development and build stack. Typically such a system can be provisioned in less than 10 minutes, thus meeting the demands of an agile development team.

```
<rpms action="install" path=
"pbl://mgr/pbl/cubit_demo1/pub/cruisecontrol-apache/">
<rpm>jdk-1_5_0_15-linux-i586.rpm</rpm>
 <rpm>cruisecontrol-bin-2.7.3-1.i386.rpm</rpm>
  <rpm>apache-beehive-1.0.2-1.i386.rpm</rpm>
 <rpm>db-derby-10.4.2.0-1.i386.rpm</rpm>
 <rpm>apache-tomcat-5.5.27-1.i386.rpm</rpm>
  </rpms>
```

Step 4: How to use CUBiT Web Services to automate cloud provisioning for testing services

CUBiT has available an extensive set of REST based API's that allow developers to automate what would normally be complicated procedures. For example, you can dynamically provision build systems in the cloud, then destroy those systems once test has completed. This is very useful for testing milestone builds that require a "clean" system in a known good state.

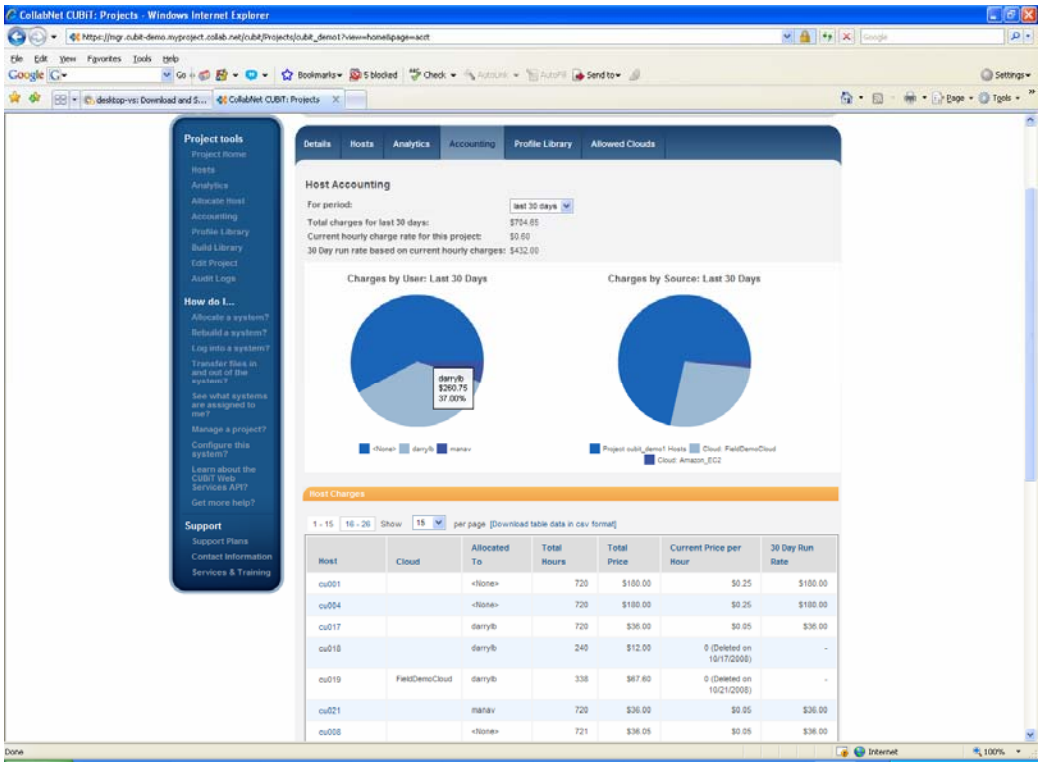
CUBiT API	*CloudCreateHosts*	*HostDelete*
Description	Create one or more hosts from a cloud given the host-type and size. The cloud will match the host-type and size to one (or more) source(s) from the cloud in order to fill the entire order.	Delete one or more hosts. The host must not be in the Immutable state for this method to work.
Requirements	Any user who is a member of a project that has been given permission to allocate from this cloud can create instances.	• CUBiT Domain Admins can delete any host, regardless if it is physical, virtual, or from a remote cloud • Users who are not CUBiT Domain Admins can delete virtual guests if the parent host is allocated to them • Users who are not CUBiT Domain Admins can also delete remote cloud hosts allocated to them
Example	`cubit_api_client.py --api-url=http://mgr.cubit-demo.myproject.collab.net/cubit_api/1 --api-user=userid --api-key=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx -s allocate_hosts_from_cloud alloc_hours=2 cloud=FieldDemoCloud host_type=basic_i386 profile=WindowsXPPro project=cubit_demo1 size=small userid=userid`	`cubit_api_client.py --api-url=http://mgr.cubit-demo.myproject.collab.net/cubit_api/1 --api-user=userid --api-key=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx –s delete_host hosts=cu001.cubit-demo.myproject.collab.net userid=userid` --force

Step 5: Determining how your cloud resources are being used and charge back

Finally, CUBiT gives teams and managers the ability for each system or profile to be used on an allocated/hour basis. Even if this feature is not used to charge developments teams, it is still valuable for determining the usage and allocation patterns of your resources.

As shown in Figure 8, CUBiT will graphically present in either a pie chart or table exactly who is using the resources in any given project and their associated costs. This ability will become more important as companies begin to dynamically allocate computing resources across projects which requires managing their own virtual private clouds.

Figure 8



Conclusions

Software development in the cloud brings adaptability and flexibility to any size project team. By utilizing some common software development tools and a new one that introduces the concept of cloud management and development services, teams can build their own cloud development environment with continuous integration automation and 'roundtrip' feedback to provide on-demand access and visibility across the development lifecycle.