CTL

distributed control dispatching framework

Introduction

by: Alex Honor, Project Leader - Open Source Software Development | ControlTier Inc.

April 21, 2008

What is CTL?

- New open source software project providing a cross-platform command dispatching framework for automated distributed management
  - Good for system *and* application administration
  - Based on the central concept of a “command dispatcher”
  - Familiar script-centric paradigm
  - Includes out of the box utilities
  - Develop your own modules in multiple scripting languages
  - “Value-adding” server applications are in the works that scale CTL

What problems does CTL try to solve?

- Avoid the age-old “looping script” for distributed management. Why?
  - Inflexible and cumbersome
  - Promotes the “ball of mud” anti-pattern
  - Mixes up coordination logic with task-specific procedure
  - Usually not rigorously written because they’re seen as temporary solutions
  - Hard to scale up

Example looping script pitfalls

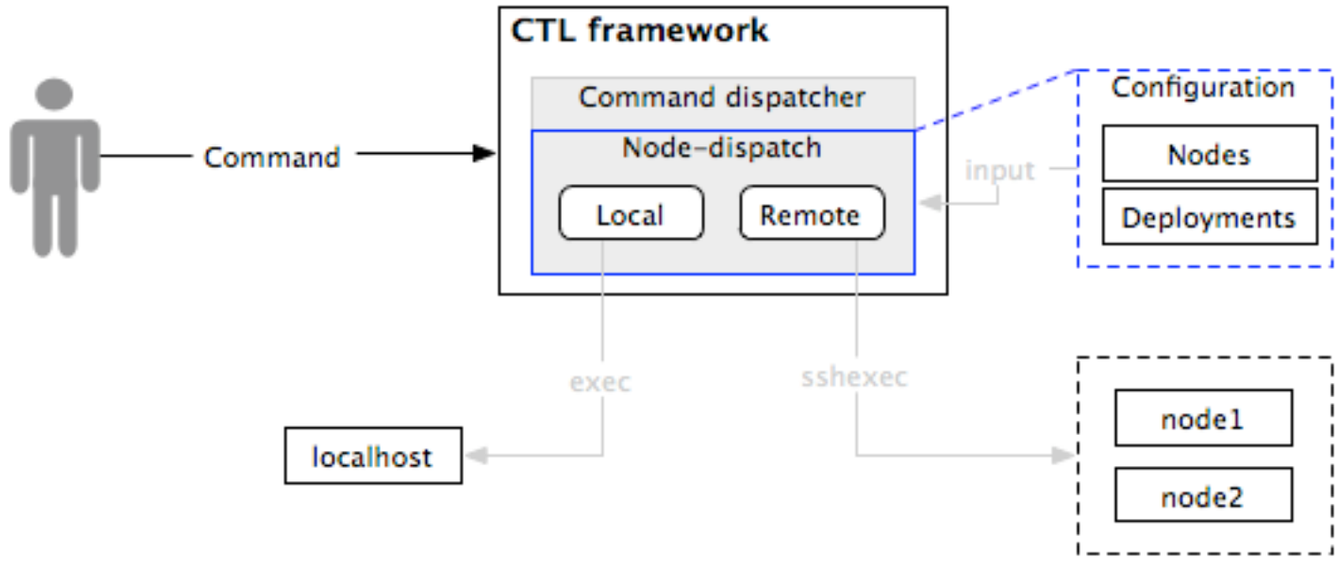
- Most of the code has nothing to do with the task at hand (4 lines to >100)
- Too many hard coded values
- Error management is very rudimentary
- Too much data and code mixed together
- Important data is buried as user variables
- Difficult to flexibly classify sets of hosts

So what's the alternative?

```
#!/bin/bash

PROG=`basename $0`
USAGE="
Usage: $PROG -g group
"
# check that an option has an argument
arg_syntax_check() {
   [ "$1" -lt 2 ] && syntax_error
}
# print USAGE and exit
syntax_error() {
   echo "$USAGE" >&2
   echo "$SYNTAX $*" >&2
   exit 2
}

while [ "$#" -gt 0 ]; do
   OPT="$1"
   case "$OPT" in
      # options with arguments
      -g)
         arg_syntax_check "$#"
         ARG_GROUP="$2"
         shift
         ;;
      --)
         shift
         break
         ;;
       # unknown option
      -?)
         syntax_error
         ;;
       # end of options, just arguments left
      *)
         break
   esac
   shift
done

#
# list of admin nodes
ADMINS="strongbad"
#
# list of web nodes
WEBS="centos development"
#
# list of all nodes
ALL="$WEBS $ADMINS"

#
# set the list of nodes to target
case "$ARG_GROUP" in
   admins)
         NODES=$ADMINS
         ;;
   webs)
         NODES=$WEBS
         ;;
   all)
         NODES=$ALL
         ;;
   *)
         break
esac

for host in $NODES
do
   if [ "$host" = development ]
   then
         sshuser="demo"
   else
         sshuser=$USER
   fi
   ssh $sshuser@$host curl http://strongbad:8080/webdav/
default/httpd.conf -o /etc/httpd/conf/httpd.conf
   if [ "$?" != 0 ]
   then
         echo "Error occured while updating httpd.conf
$host"
         exit 1
   fi
   ssh $sshuser@$host /usr/sbin/apachectl configtest
   if [ "$?" != 0 ]
   then
         echo "Error occured checking configuration file on
$host"
         exit 1
   fi
   ssh $sshuser@$host sudo /usr/sbin/apachectl restart
   if [ "$?" != 0 ]
   then
         echo "Error occured while restarting httpd on $host"
         exit 1
   fi
   ssh $sshuser@$host /usr/sbin/apachectl status
   if [ "$?" != 0 ]
   then
         echo "Bad status on $host"
         exit 1
   fi
done
```

What is Dispatching?

- Takes a command and invokes it locally or remotely and optionally, in parallel
- Network abstraction provided via “node dispatch”
- Centralized configuration maintains metadata profiles of nodes and deployments



- Two kinds of command dispatching available
  - Ad-hoc commands: Anything you normally type at the shell
  - Defined commands: Commands saved into a library called a module

How about some examples?

Execute ad-hoc commands via “ctl-exec”

- Usage: ctl-exec [*args*] [-- *command*]
- Check if http is running on all machines
  - ctl-exec -- ps | grep httpd
- Use filtering keywords to target specific hosts
  - -I (inclusion), -X (exclusion)
  - keywords: hostname, os-name, os-family, os-version, os-arch, tags
- Use -threadcount and -keepgoing flags to run across large numbers
- Without *command*, the matching nodes are listed

Why is that cool?

ctl-exec lets you run ad-hoc commands through “command dispatching”

ctl-exec separates dispatch from task specific procedure

Node dispatching provides a level of network abstraction

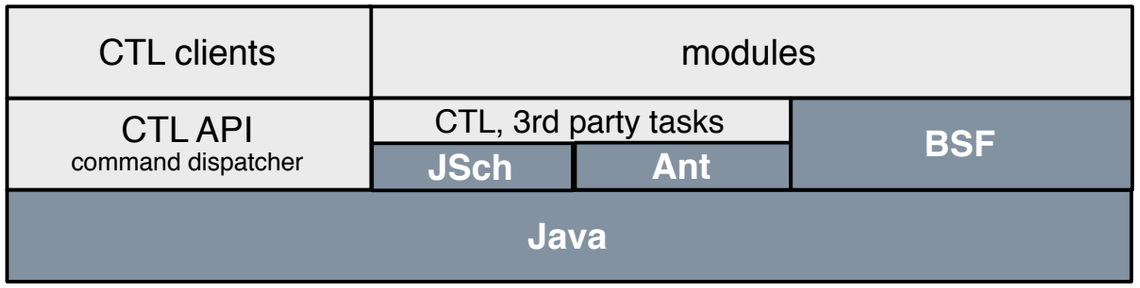
Defaults to sequencing but threading allows you to run commands in parallel

Node metadata provides a flexible set of keywords to target actions

Consider ctl-exec the next time you’re about to write a looping script.

Under the hood

CTL Software architecture:



- Based on Java (currently testing 1.5 but eventually support 1.4 and above)

- Remote invocation done over SSH2 via JSch (Java Secure Channel)

- Apache Bean Scripting Framework and Ant provide support for multiple implementation languages

- Can be embedded in your existing management frameworks or write scripts around CTL

Deploying CTL

Assumptions we make

- Passive (sshd is the active process)
- Add admin host SSH public key to target hosts authorized_keys
- Does not require root (in fact that's discouraged)
- Java Installed. 1.5 (soon 1.4+)

Installation

1. Extract the tgz/zip archive
2. Run `ctl-setup`
3. Optional: Centralized web repository. Any HTTP server will do, but WebDAV is pretty cool

More examples with ctl-exec

Scenario: Sync the apache configuration

- Step 1: Push the config
  - ctl-exec -l tags=web -- curl http://strongbad/webdav/default/httpd.conf
- Step 2: Test the config
  - ctl-exec -l tags=web -- apachectl configtest
- Step 3: Restart the httpd
  - ctl-exec -l tags=web -- sudo apachectl restart
- Step 4: Check the process
  - ctl-exec -l tags=web -- apachectl status

Roadmap for ctl-exec

- Features in the next release (all from user feedback)
  - here documents. Create multi-line scripts

```
ctl-exec -I tags=web -stdin <<END
  statement 1
  statement 2
  ...
  statement 3
END
```

  - -s,--scriptfile. `ctl-exec -s myscript.sh`
  - -q,--quiet mode. Show only error messages.
  - -R,--retry flag. Like -K,--keepgoing except it will cycle back and retry hosts where command failed
  - report generation
  - *Your suggestions here!*

Defined Commands

Define and execute commands in a module

What problems do defined commands solve?

- Lack of standardized function library for critical operational tasks
- Huge monolithic scripts that are inflexible, hard to debug and do not lend themselves to unit testing!
- Uncontrolled packaging, versioning and distribution of important automation code.
- Weak or non-existent security model
- Missing or inconsistent data model to drive automation code
- *A full featured framework that supports script writers to develop, deploy and operate online service environments*

Defined commands

- CTL allows you to take your existing script code and save it as a defined command in a library called a module.

- These commands then become exposed to the CTL framework and available to the command dispatcher.

- The command dispatcher can then invoke your commands with network abstraction via node dispatch

- You can define a data model to drive your commands

- Other framework benefits include advanced features like object-orientation features, access control and reporting

How about some examples?

Defined commands run via “ctl”

- Usage: ctl -m *module* -c *command* [-- [args]]
- The “-X” and “-I” flags turn on node dispatching
- Check if http is running on all machines except strongbad
  - `ctl -X strongbad -m shellutil -c ps -- -pname httpd`
- Check if sshd is running on all Linux machines (sshd already assumed)
  - `ctl -I os-name=Linux -m shellutil -c ps -- -pname sshd`
- Without “*-c command*” the matching commands are listed

Why is that cool?

No more monolithic large and unweildly scripts. They decompose to modular functions available for reuse

Node dispatching provides network abstraction, sequencing or concurrency to all your procedures

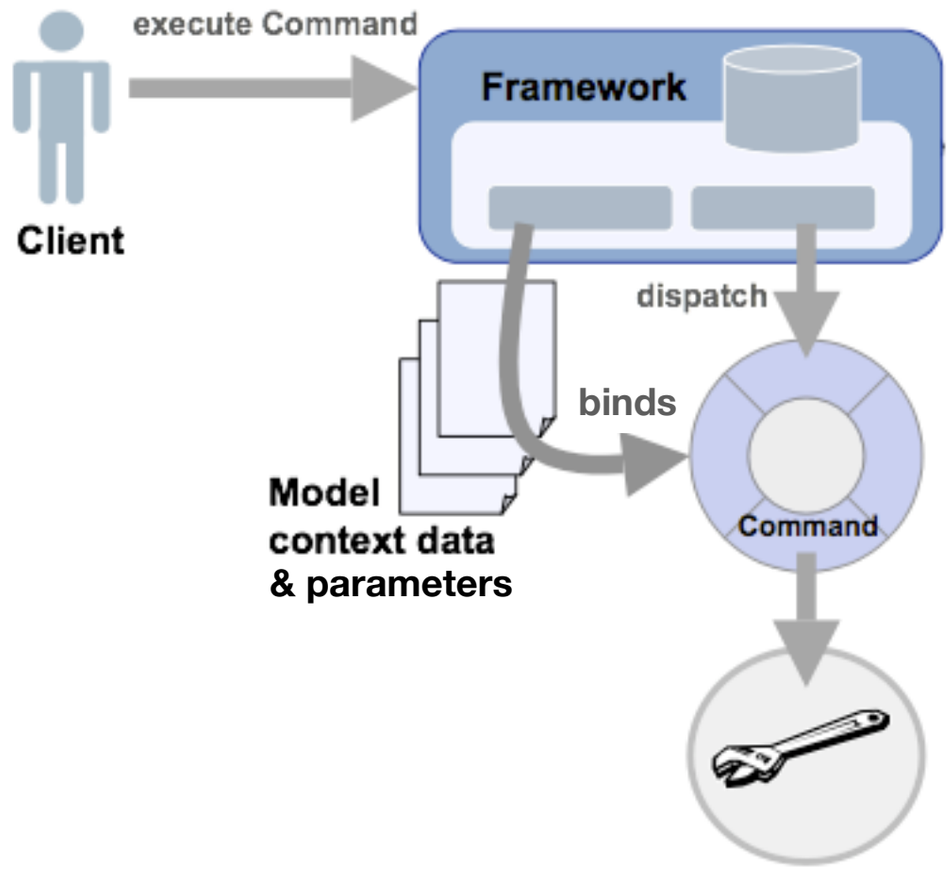
CTL modules lets you hide command implementations, combine them into sequences and allows you to mix scripting languages.

CTL provides a standard way of packaging, distributing, deploying and executing new modules.

Under the hood

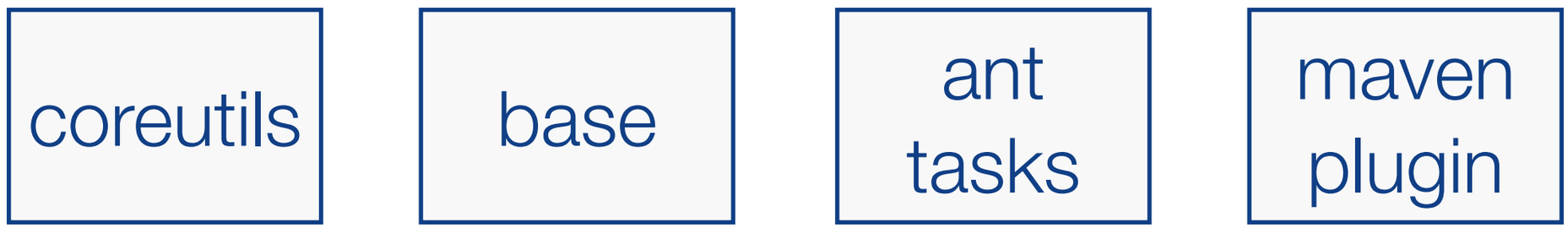
How module dispatching works:

- Framework takes command request,
  - resolves it to a “handler”,
  - creates a data binding “context”,
  - and dispatches control to the handler
- Your command interfaces a tool or encapsulates your own procedure



Out of the box utilities

- “coreutils” Cross platform utilities inspired by the GNU coreutils.
  - fileutil, netutil, shellutil, textutil
- ProjectBuilder
  - Development tool for defining new CTL controller modules
- Ant tasks and Maven plugin
  - Enables Java build life cycle to extend into deployment and operation!



Time for more examples...

- Scenario: You need to push out a new version of Apache httpd.conf, restart the httpd processes and then check their status

- Assumption: You want to do this centrally

  - You'll maintain httpd.conf files in a central repository

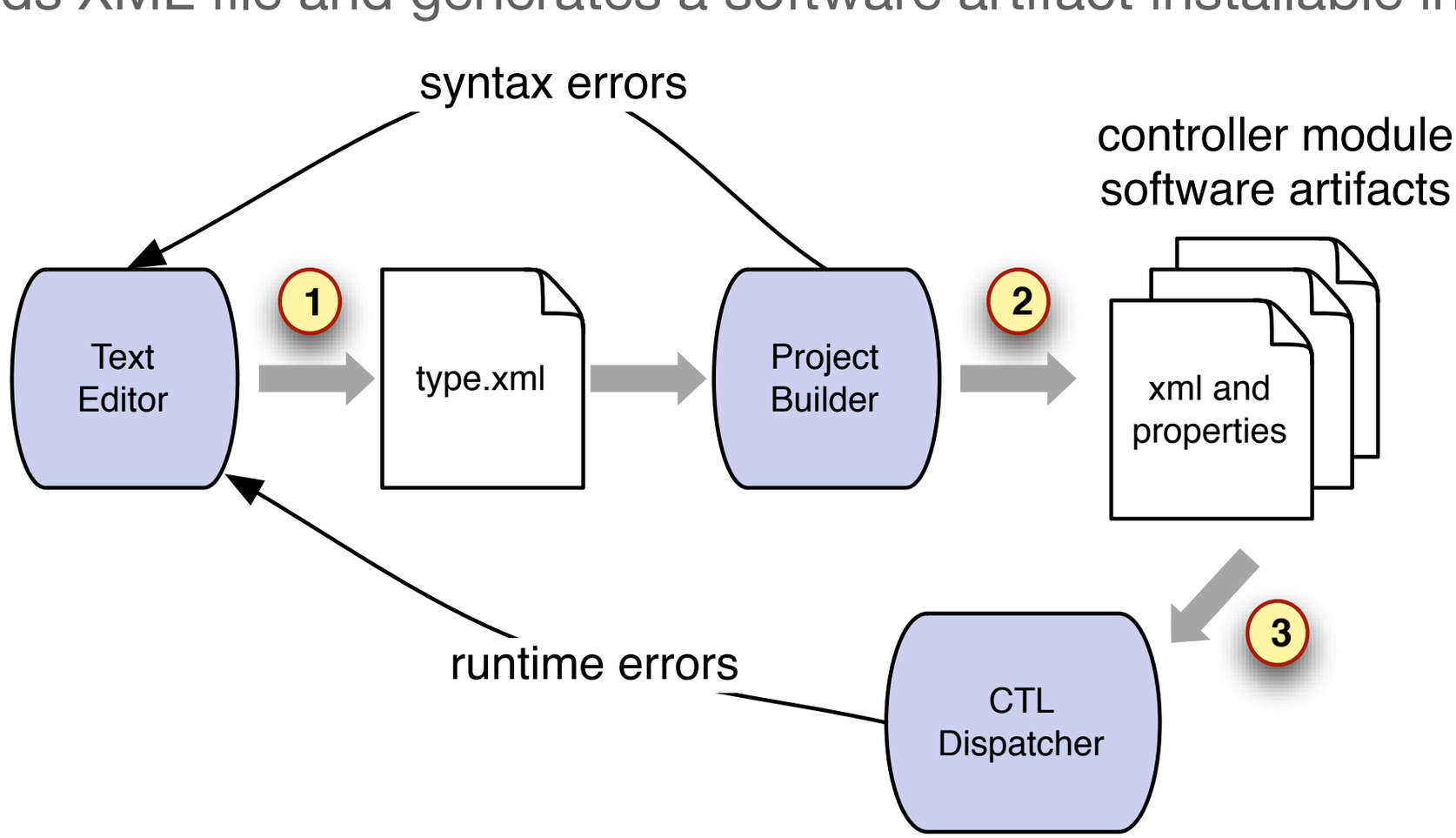
- You have multiple nodes with Apache deployments

- Based on the ad hoc commands shown earlier with `ctl-exec`

- Capture them as "defined commands" and run via `ctl`

Defining new controller modules is simple

- Controller modules are defined in an XML file
  - Contains a set of command and attribute definitions
- Controller modules are built via ProjectBuilder utility
  - reads XML file and generates a software artifact installable in CTL



The “type.xml”

- Defines commands
  - Choose your command type and plug in your implementation
  - Define any needed options
- Defines attributes
  - Creates a data model for your command
  - Use it to default your command parameters

ctl -m apacheutil -c status -- -port 80

```
<type name=”apacheutil”>
  <attributes>
    <attribute-default
          name=”port” value=”80”/>
  </attributes>
  <commands>
   <command name=”status”>
    <execution-string>sh</execution-string>
    <argument-string>
     netstat -an | grep ${opts.port}
    </argument-string>
    <opts>
      <opt parameter=”port” type=”string”
            property=”opts.port”
            defaultproperty=”entity.attribute.port”/>
    </opts>
   </command>
  </command>
</type>
```

Example: Define commands in a module

Goal: Create a module that defines apache utility commands

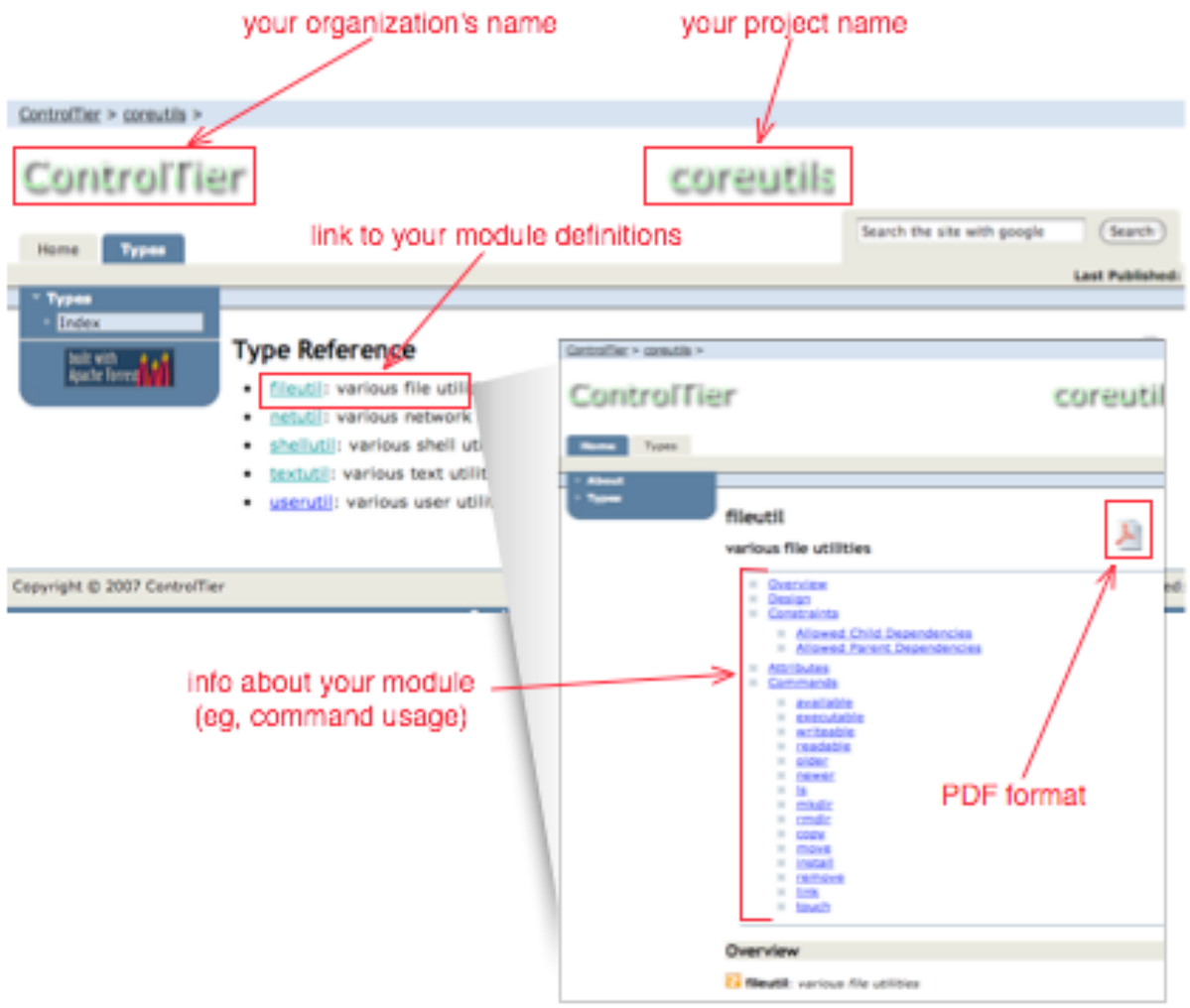
- Step 1: Create and edit
  - ctl -m ProjectBuilder -c create-type -- -type apacheutil
- Step 2: Build and deploy it
  - edit deployments.properties
  - ctl -m ProjectBuilder -c build-type -- -type apacheutil -deploy
- Step 3: Run it
  - ctl -m apacheutil -c status

Use the right language for the job at hand

- CTL supports several kinds of command handlers:
  - Bean Shell Framework: python, ruby, javascript, groovy, and others
  - Shell: Unix shells, Windows .bat, Perl
  - Ant: Ant tasks and types
  - Workflows: Define sequences of commands (or other workflows) along with success and error handling, notification and reporting
  - Or reuse defined commands from “coreutils” library (and eventually reuse your own)

Runbook (generated docs)

- Your “type.xml” file is used to generate web documentation
  - Gives usage info for your commands
- Great for handing off to others
- Always current with your modules
- Based on Apache Forrest



Distribute your module

- Declare deployments in the deployments.properties file
  - maps modules to nodes
  - used by nodedispatch in “`ctl`”
- Use `ctl-depot` to deploy modules
  - ctl-exec -- ctl-depot -D

Example: sync'ing the Apache configuration

Use "apacheutil" to run the defined commands

- Synchronize the configuration file
  - ctl -l tags=web -m apacheutil -c configget
- Test the configurations
  - ctl -l tags=web -m apacheutil -c configtest
- Restart the httpd processes
  - ctl -l tags=web -m apacheutil -c restart
- Check status:
  - ctl -l tags=web -m apacheutil -c status

Each command encapsulates its implementations and presents itself for future reuse.

Workflows: Command sequences

- Workflows call a sequence of previously defined commands.
- Can run command sequences in multiple threads
- The error handling options:
  - run another command
  - fail
  - report
  - prompt the user
  - email

```
<command name="sync-config"
    description="sync and restart httpd"
      command-type="WorkflowCommand"
      is-static="true"
      error-handler-type="FAIL"
      >
 <workflow threadcount="1">
   <command name="configget"/>
   <command name="configtest"/>
   <command name="restart"/>
   <command name="status"/>
 </workflow>
</command>
```

Example: Config syncing via the workflow

- Run the syn-config command
    - ctl -l tags=web -m apacheutil -c sync-config

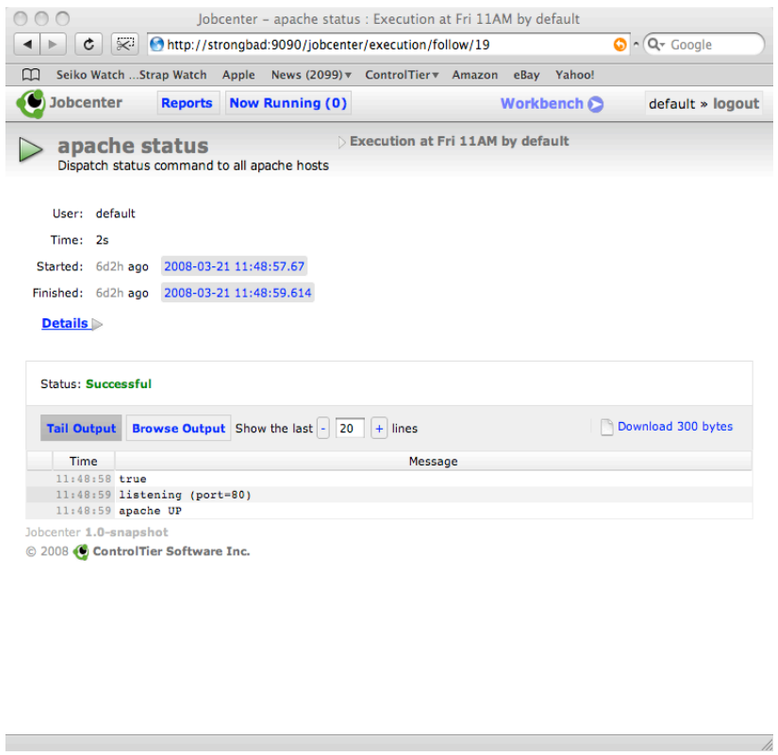
The “sync-config” workflow is itself a command that can be used within another workflow.

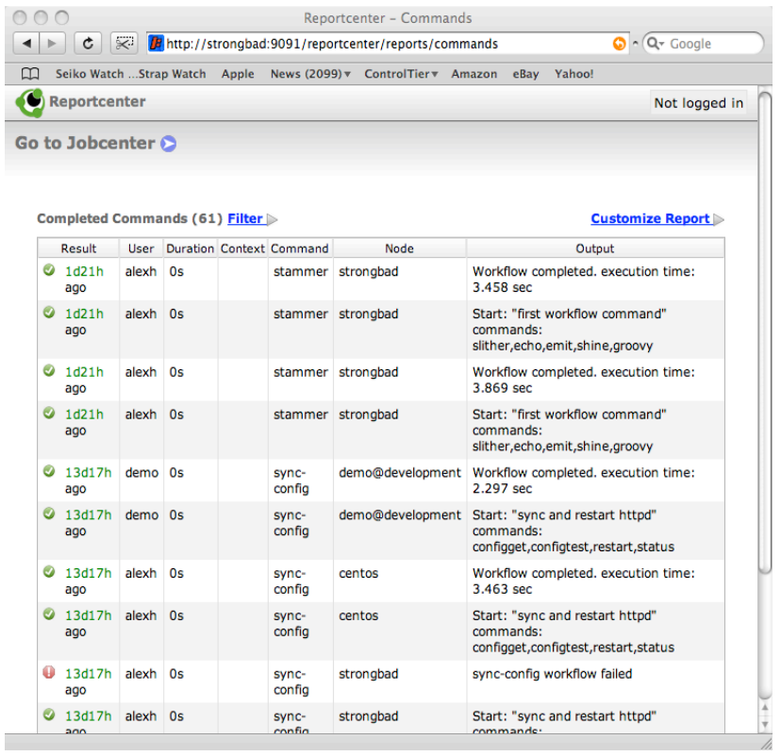
Server applications

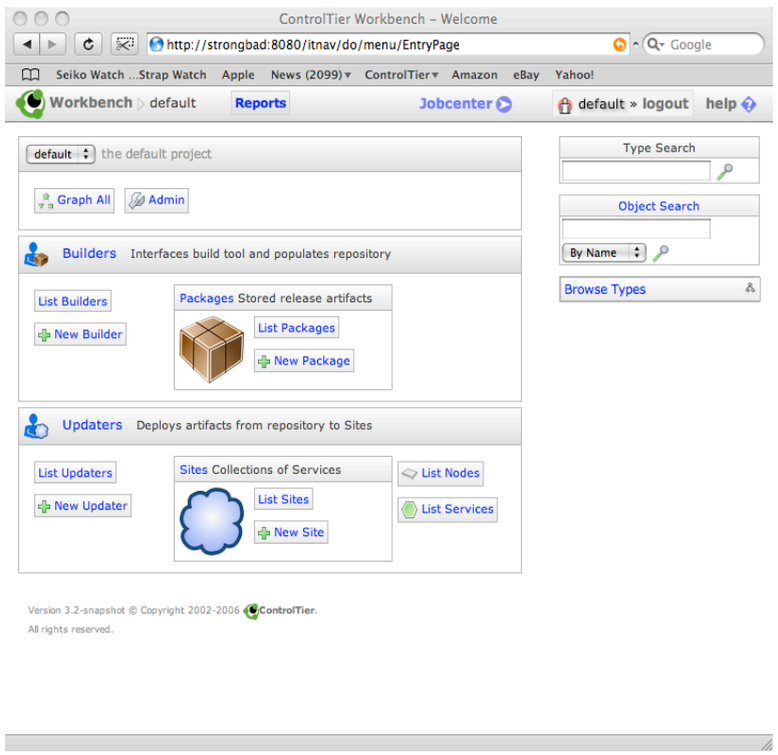
Lets you scale CTL

“Value-adding*” server applications in the works

- Jobcenter: Scheduled command execution:
- Reportcenter: Logging and reporting of commands:
- Design Workbench: Integrated model of all controller definitions:





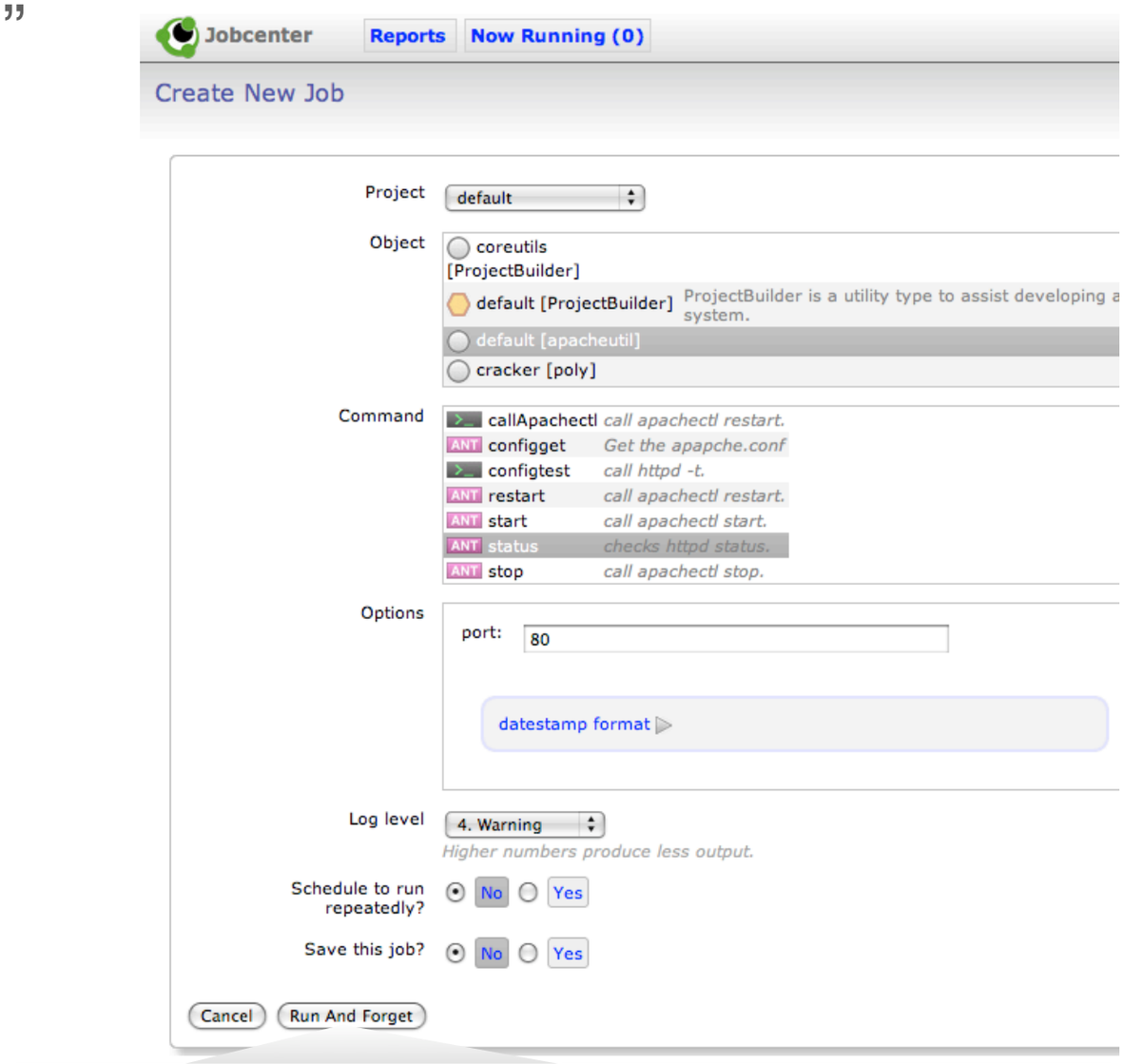


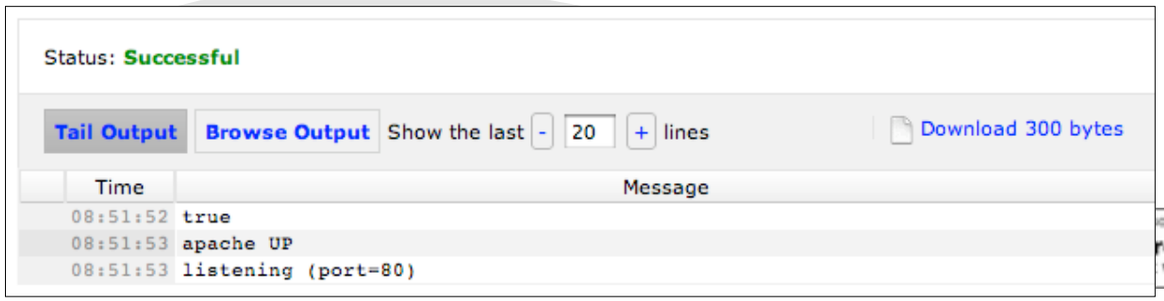
* All these applications are also open-source

Sponsored by ControlTier software

Jobcenter: Web-based self service

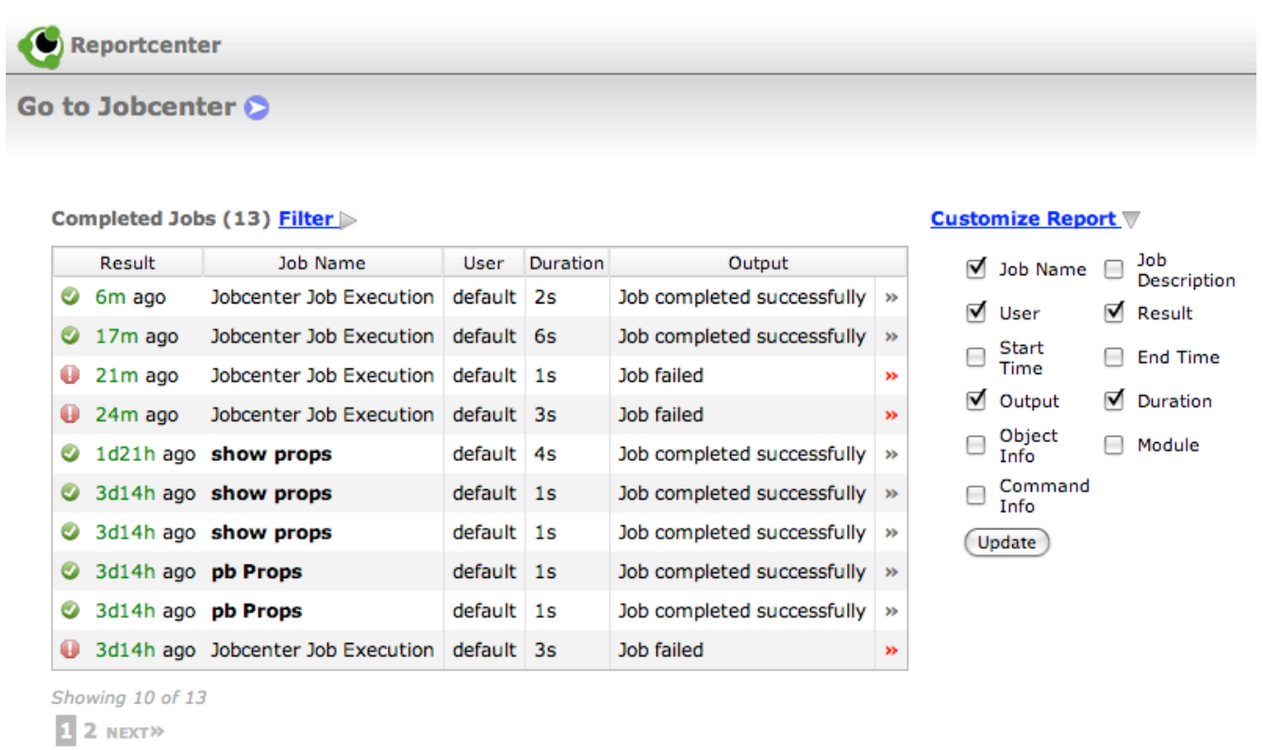
- Exposes defined commands as “jobs”
- Jobs can be
  - scheduled to run repeatedly,
  - saved or
  - run and forgotten.





Reportcenter: Centralized Reporting

- Adds logging and auditing capability to CTL
- Good for keeping track of activity in larger CTL environments
- Based on Log4J. Has a listener that receives logging requests with populated MDC fields
- Being made open ended to support many kinds of “report activities”



Parting thoughts

- CTL built on the idea of a dispatcher that supports network abstraction and concurrency
- Dispatcher simplifies your scripts by separating centralized dispatching logic from task-specific procedure
- Introduces node and deployment metadata that can be used as filtering keywords and as a standard method to target actions
- Supports ad-hoc *and* defined commands. Defining commands is simple and can be done in any mixture of scripting languages.
- Several enterprise class server apps on the way to help you use CTL in large scale environments.
- This is a new project so your feedback is welcomed and encouraged!

CTL

distributed control dispatching framework

Resources

Documentation
http://ctl.controltier.com

Comments and questions on Google Group
http://groups.google.com/group/controltier