# CFENGINE IN A DAY

## NICK ANDERSON

# BEFORE WE START

# SCHEDULE

A rough schedule (this varies greatly).

| Time | Activity |
|---|---|
| 09:00 AM - 10:30 AM | Class |
| 10:30 AM - 11:00 AM | Morning break |
| 11:00 AM - 12:30 PM | Class |
| 12:30 PM - 01:30 PM | Lunch break |
| 01:30 PM - 03:00 PM | Class |
| 03:00 PM - 03:30 PM | Afternoon break |
| 03:30 PM - 05:00 PM | Class |

Who will volunteer to **remind me of breaks**?

# USE THE SOURCE

The source materials for this presentation can be found on github.

- https://github.com/nickanderson/cfengine-training

See a **mistake**? Please consider submitting a Pull Request.

# HI, I'M NICK

- Wife, 2 kids and a dog
- Sysadmin/Infrastructure Engineer/Doer of things
- Find me online
  - nick@cmdln.org | nick.anderson@cfengine.com
  - @cmdln_
  - cmdln.org
  - linkedin.com/in/hithisisnick
  - `9274 E588 E866 A10B 713C 9CCD 9EB3 AD42 5D1C CC11`

## WHO ARE YOU?

- Name
- Role
- Goals for this training

# WHY AUTOMATION?

*Every time someone logs onto a system by hand, they jeopardize everyone's understanding of the system. – Mark Burgess*

# WHAT IS CFENGINE?

Modern, Secure, scale-able, and agile **infrastructure automation tool** designed to provide continuous operations using a distributed model based approach.

# HISTORY

- Written by Mark Burgess
- Originally released in 1993.
- Computer Immunology (Self Healing) 1998
- CFEngine 2 (1998)
- Promise Theory (2005)
- CFEngine 3 (2008)
- Company Founded (2008)
- CFEngine 3.10 (2016)?

# SMALL

As of October 2016:

- ~ 125k Lines of code
- ~ 20M Memory usage
- ~ 10M Package size

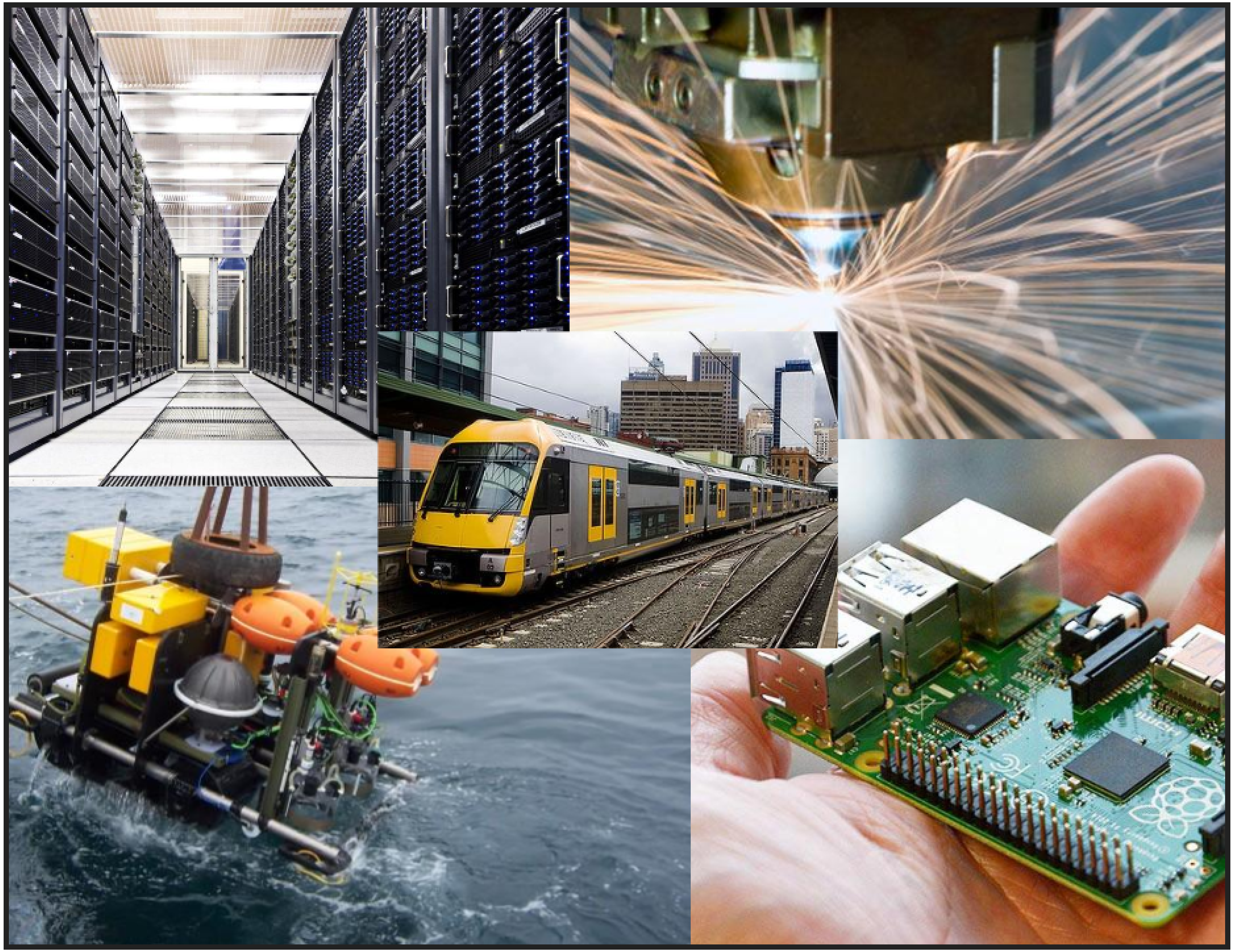
## SECURE

A good track record.

- http://web.nvd.nist.gov/view/vuln/search

# PORTABLE

Known to run on many platforms, and interesting places.

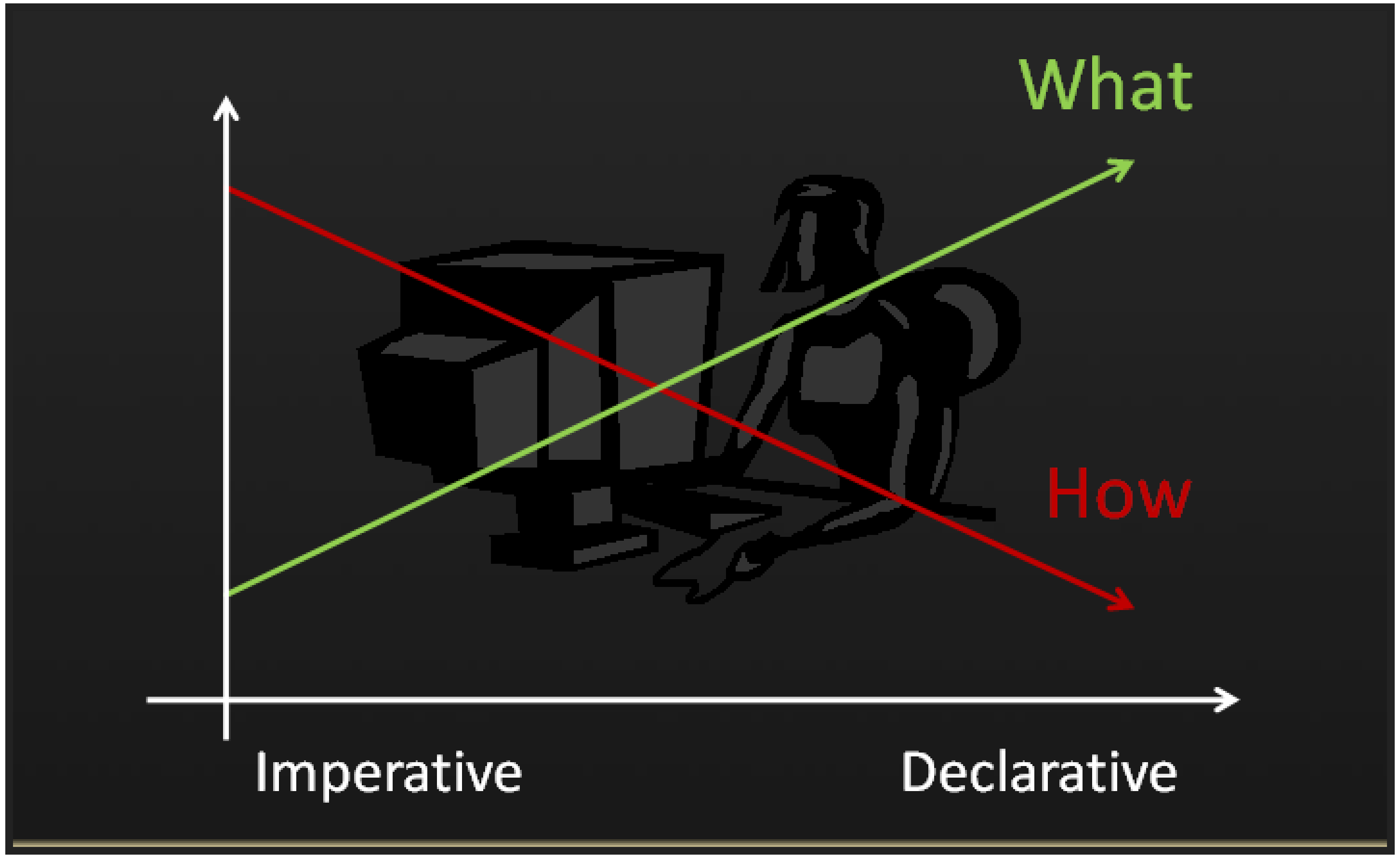


# RESILIENT

Image: Scene from Monty Python & the Holy Grail

- Policy cached locally
- Decisions made locally
- Repair what you can and re-vist soon

# DECLARATIVE



Focus on the **desired end state**.

- Focus on the GOAL
- Align with business priorities

# DECLARATIVE VS IMPERATIVE

Strict Order of operations vs Goals/Way-points

# TYPICAL USE CASES

- Any **local** operation
  - Manage files, processes, commands
- Compliance
  - Security Hardening
- Application Deployment/Management
- Inventory
- What's possible?

# EDITIONS

# COMMUNITY (CORE)

- Github
- Tarballs
- Binary Packages
- Linux Package Repositories

Quick Install Community

```
wget -O- http://cfengine.package-repos.s3.amazonaws.com/\
  quickinstall/quick-install-cfengine-community.sh | sudo bash
```

# ENTERPRISE FEATURES

- Inventory Reporting
- Change Reporting
- Audit and Compliance
- Anomoly Detection
- Monitoring
- REST API
- SQL Reporting
- File Integrity Monitoring

# ENTERPRISE

- Vagrant Environment
- Binary Packages

Quick Install Enterprise

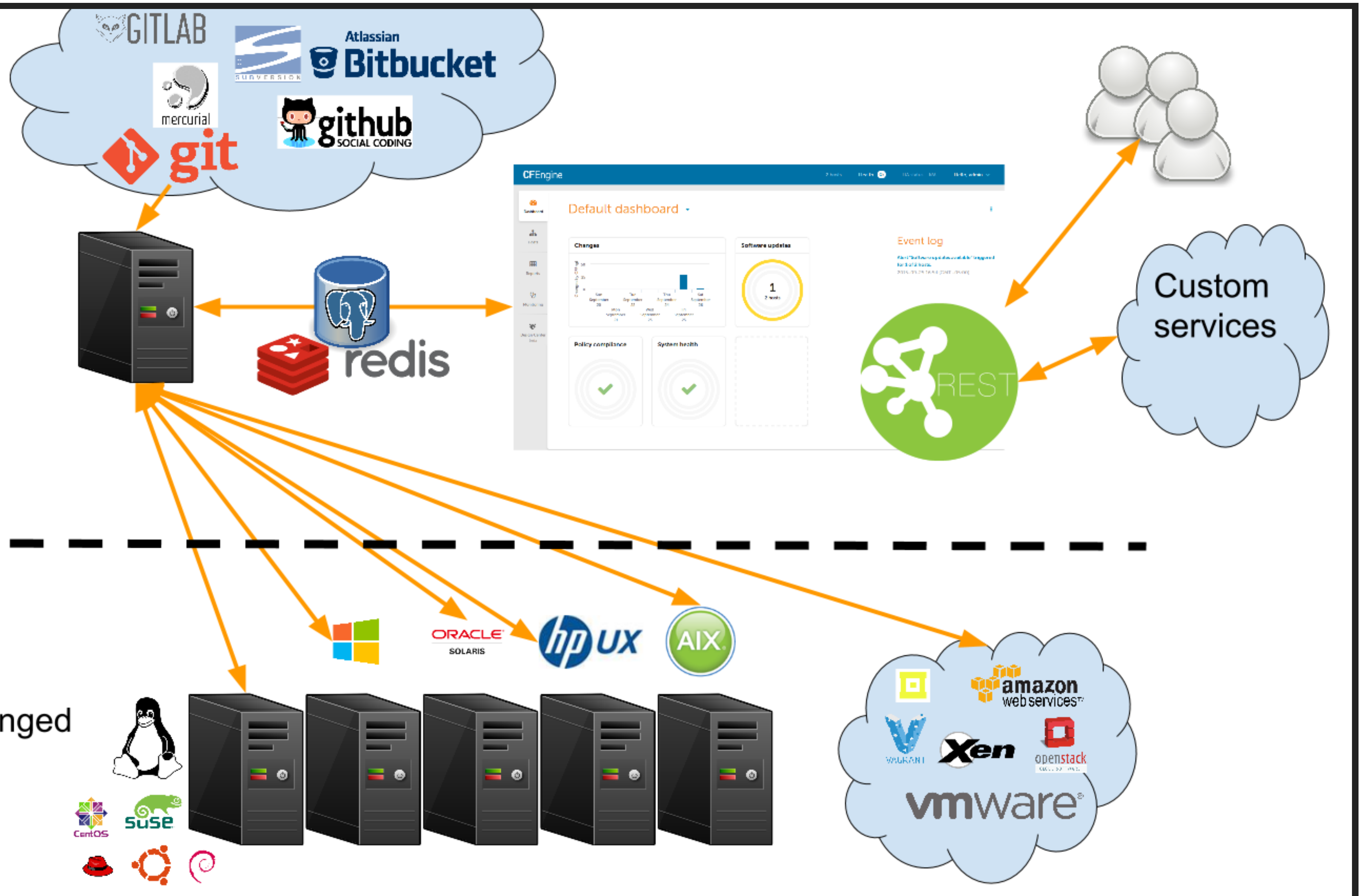
```
wget http://cfengine.package-repos.s3.amazonaws.com/quickinstall/\
quick-install-cfengine-enterprise.sh
sudo bash ./quick-install-cfengine-enterprise.sh <hub|agent>
```

# CLIENT/SERVER ARCHITECTURE

## CFEngine Policy Server

Policies **should** be maintained in a version control repository.

Policy distributed from **/var/cfengine/masterfiles**.



## CFEngine Client

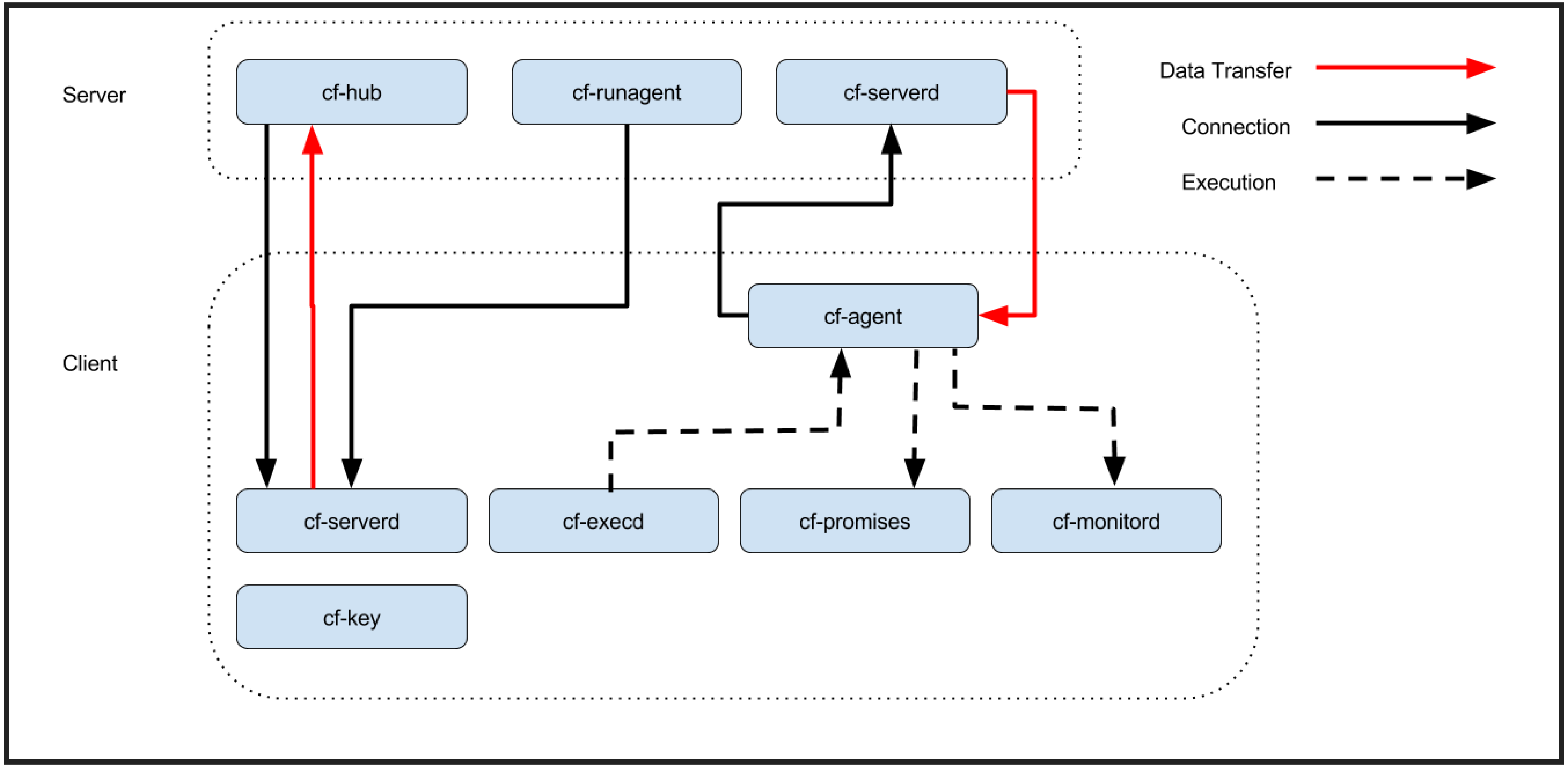
Agents update policy when **cf_promises_validated** has changed on the policy server.

# CLIENT SERVER COMMUNICATIONS

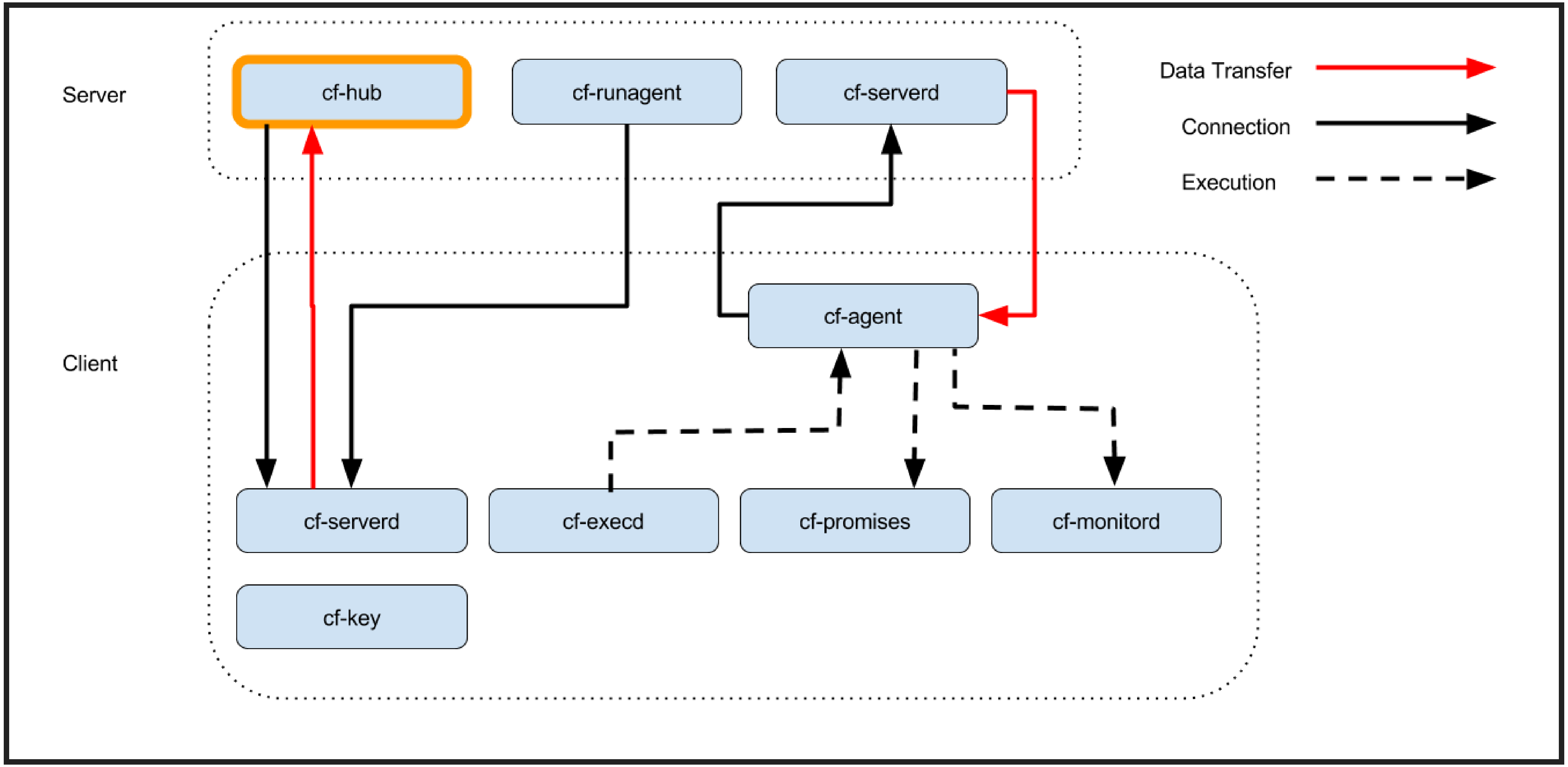
- Utilizes port 5308 for client – server communication
- Encrypted (TLS)
- SSH-like model (public/private keypair)
- Does not use complicated chains of trust
- Agents download policy from hub
- Hub downloads reports from remote agents

# AGENT COMPONENTS

# AGENT COMPONENTS



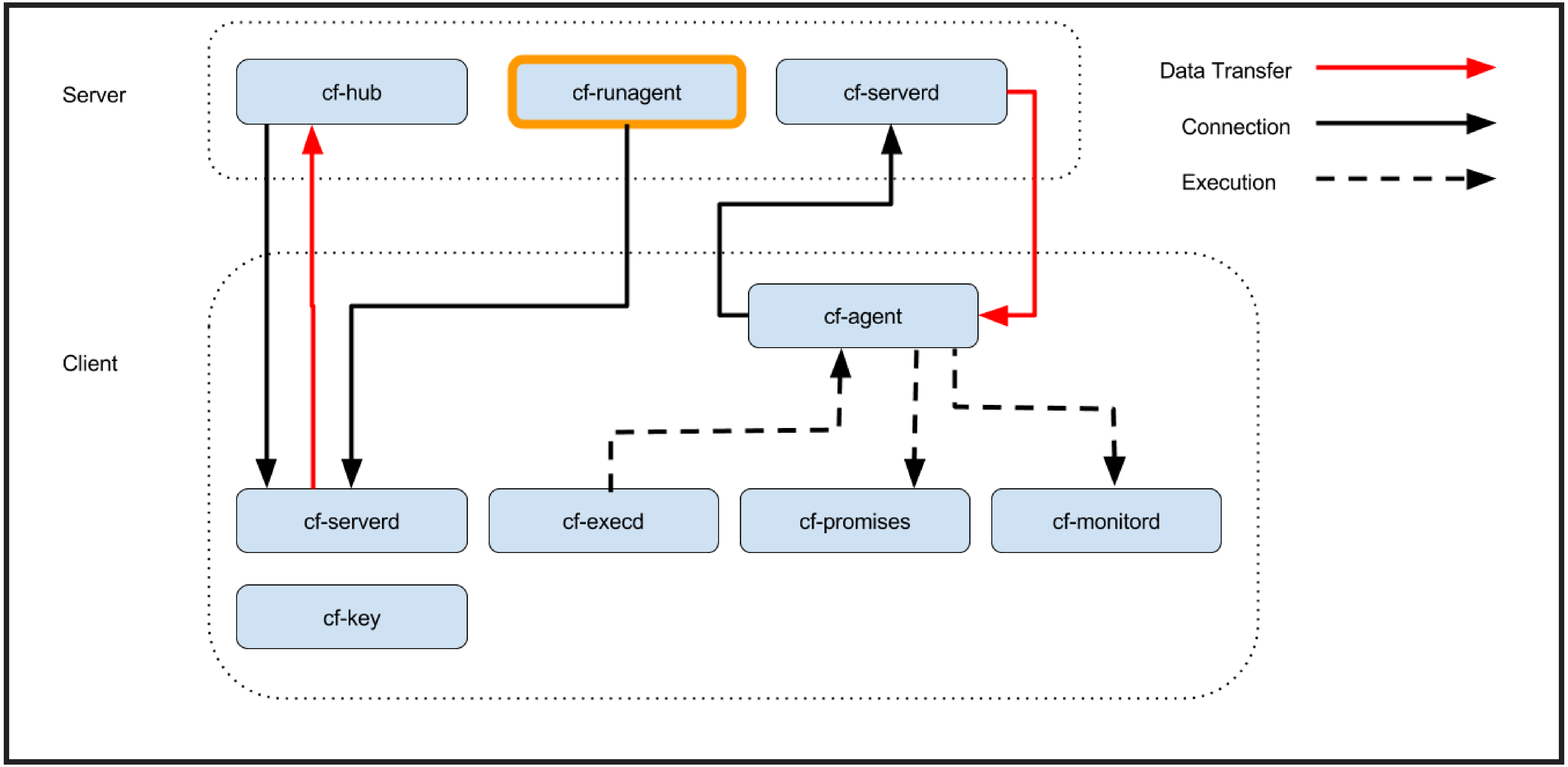
# CF-HUB


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

## `CF-HUB`

- Collect reports from remote agents
- Obeys `"common control"` and `"hub control"` bodies

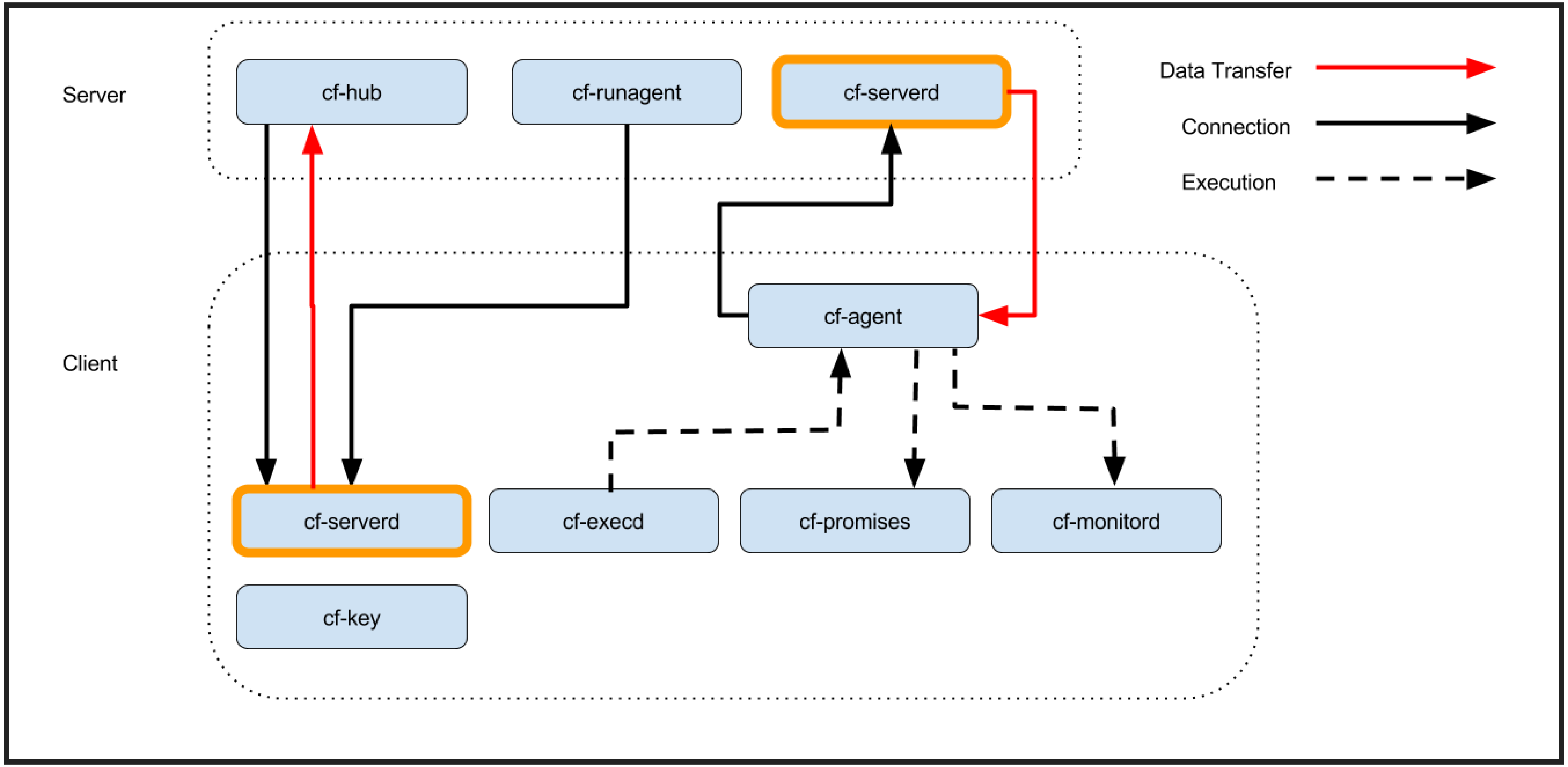
# CF-RUNAGENT


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# CF-RUNAGENT

- Invokes cf-agent on remote hosts
- Normally run on the server to “ping” the clients
- Cannot invoke arbitrary commands - just wake up cf-agent
- Define classes to modify behavior
- Specify bundlesequence (`--remote-bundles` added in 3.10)

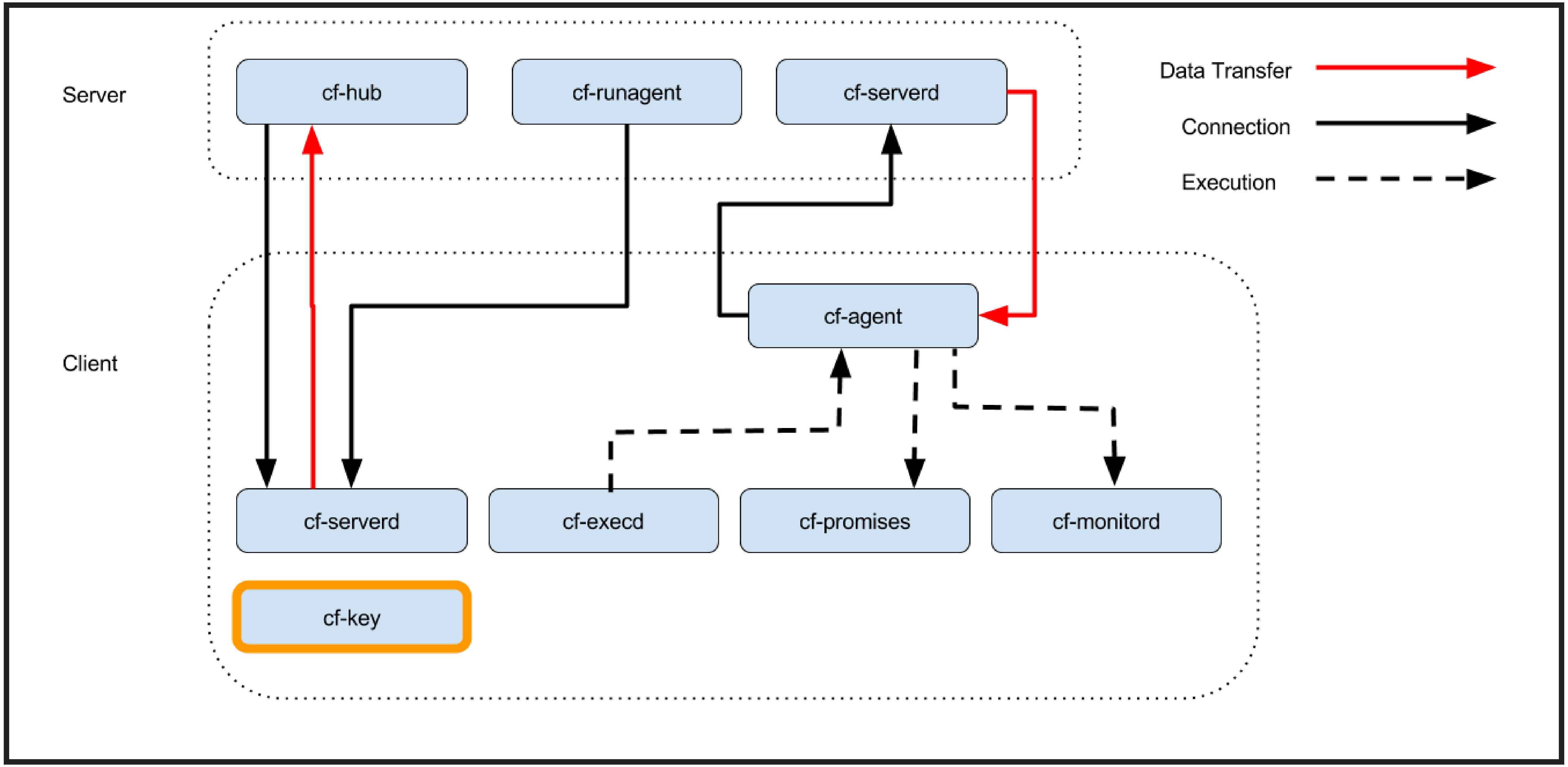
# CF-SERVERD


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# `CF-SERVERD`

- Listens for connections on **TCP/5308**
- Enforces access control and authentication
- Serves files
- Serves `cf-runagent` requests
- Serves reports to `cf-hub`
- Runs on both server and clients
- Evaluates "`common`" and "`server`" bundles
- Obeys body "`server control`"

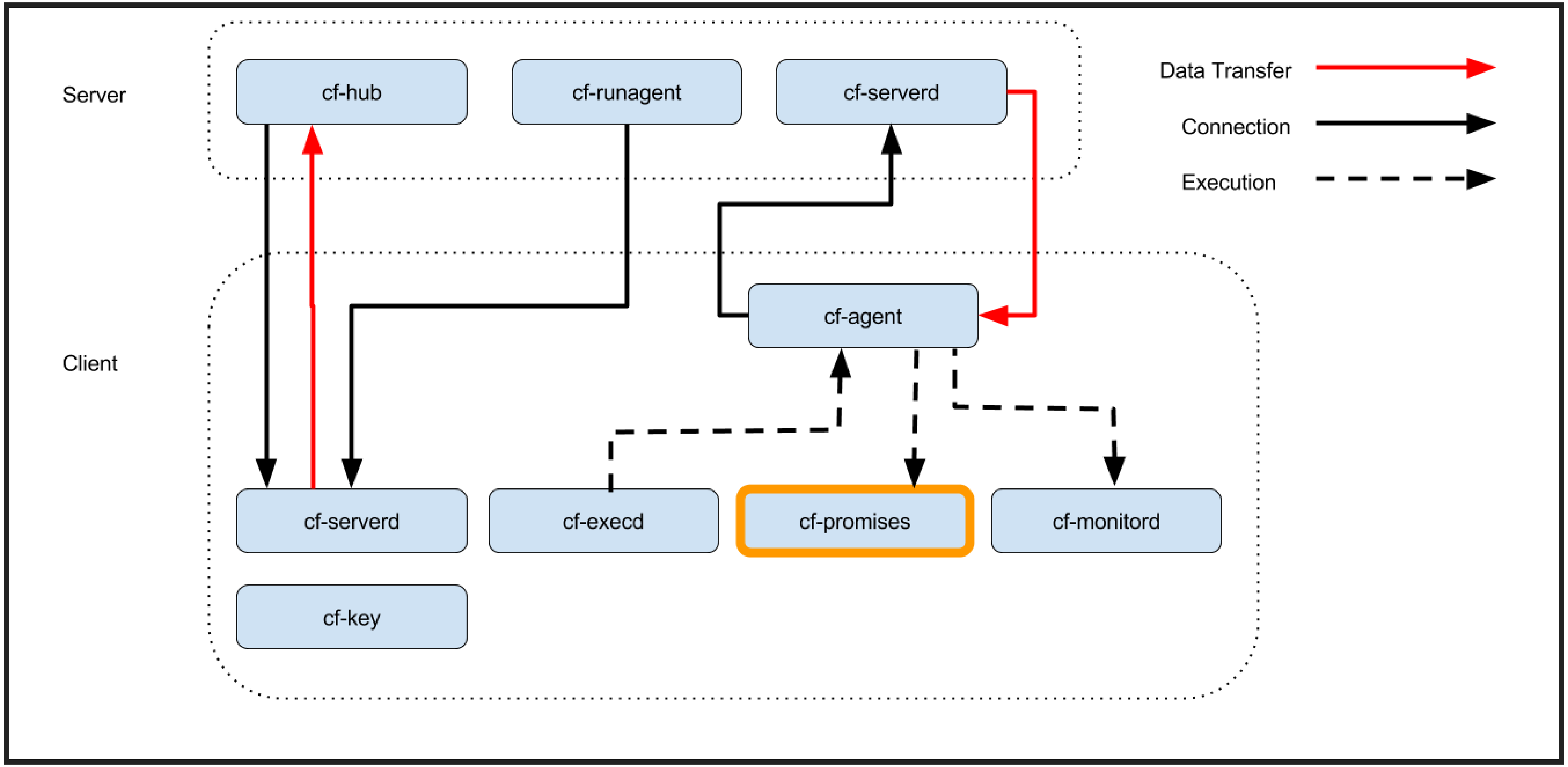
# CF-KEY


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

## CF-KEY

- Show recent connections
- Manage trust of public keys
- Generates key pairs
- Installs Enterprise License

# CF-PROMISES


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# CF-PROMISES

- Checks Syntax
- Performs pre-evaluation & discovery
- Dump Syntax
- Tag Policy Releases
- Show Variables
- Show Classes

# PRO TIP: SYNTAX CHECKING

## Full Syntax Check

```
cf-promises --eval-functions=yes --full-check \
  -f ./examples/00-01-hello_world.cf
```

- Full check requires `body common control` (or `bundle agent main`). Typically this is only used when you run cf-promsies against promises.cf or update.cf

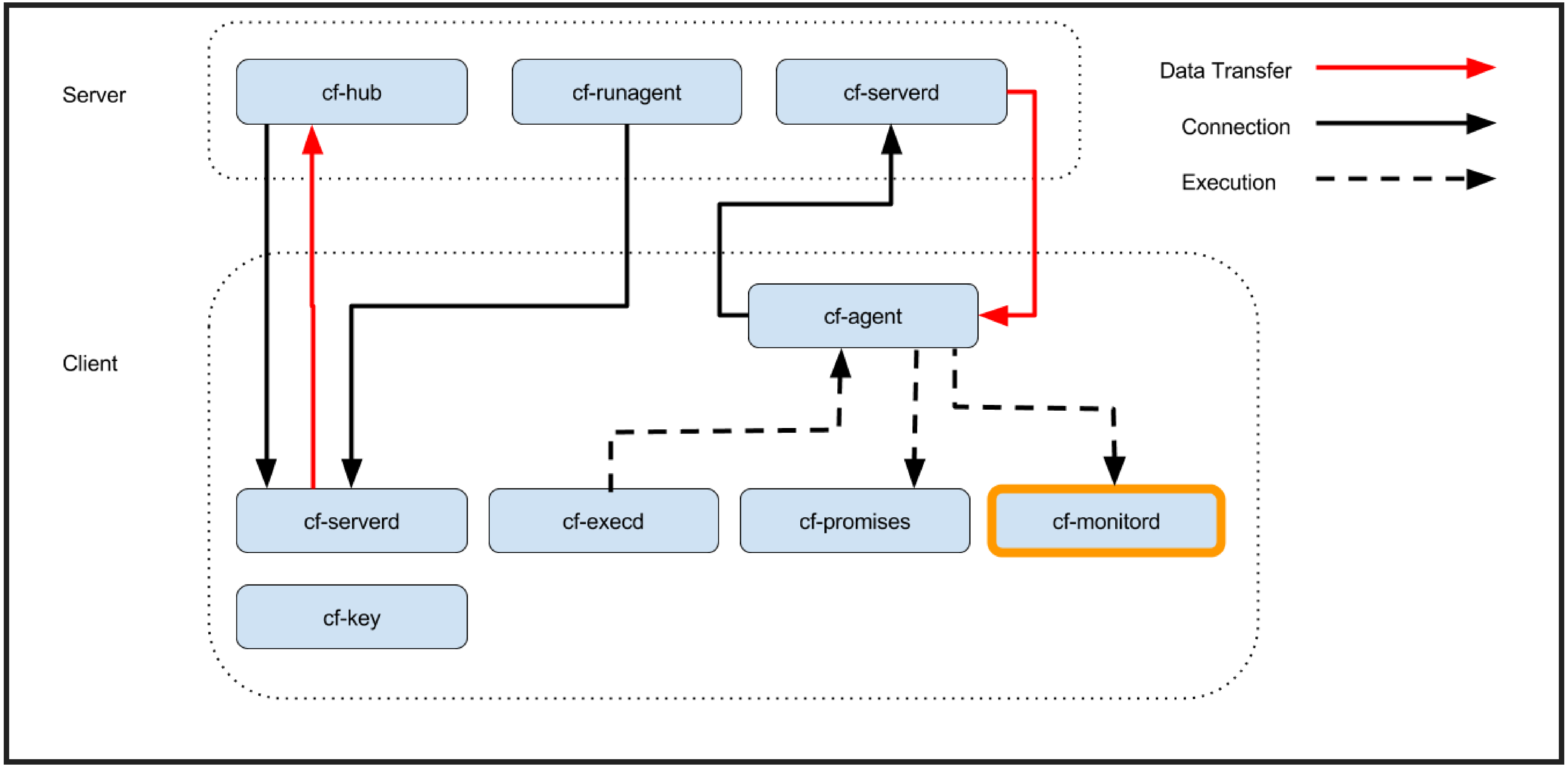
# Partial Syntax Check

```
cf-promises -f ./examples/00-01-hello_world.cf
```

## Build into your workflow!

- Editors
- vcs hooks
- build systems

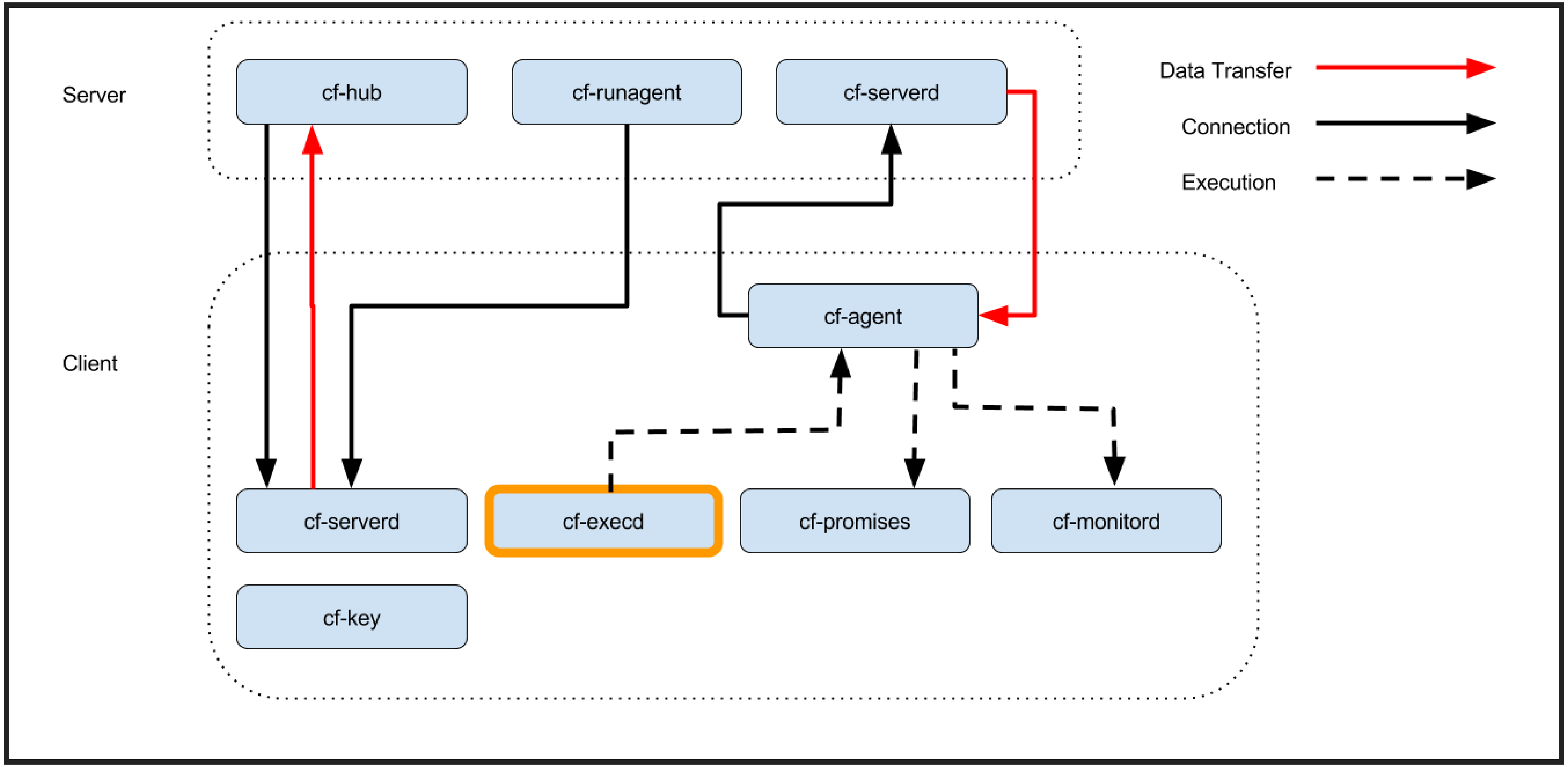
# CF-MONITORD


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# CF-MONITORD

- Collect system status information
- Provides information to `cf-agent` through `mon.*` variables
- Defines classes based on anomaly detection
- Evaluates "`common`" and "`monitor`" bundles
- Obeys "`common control`" and "`monitor control`" bodies

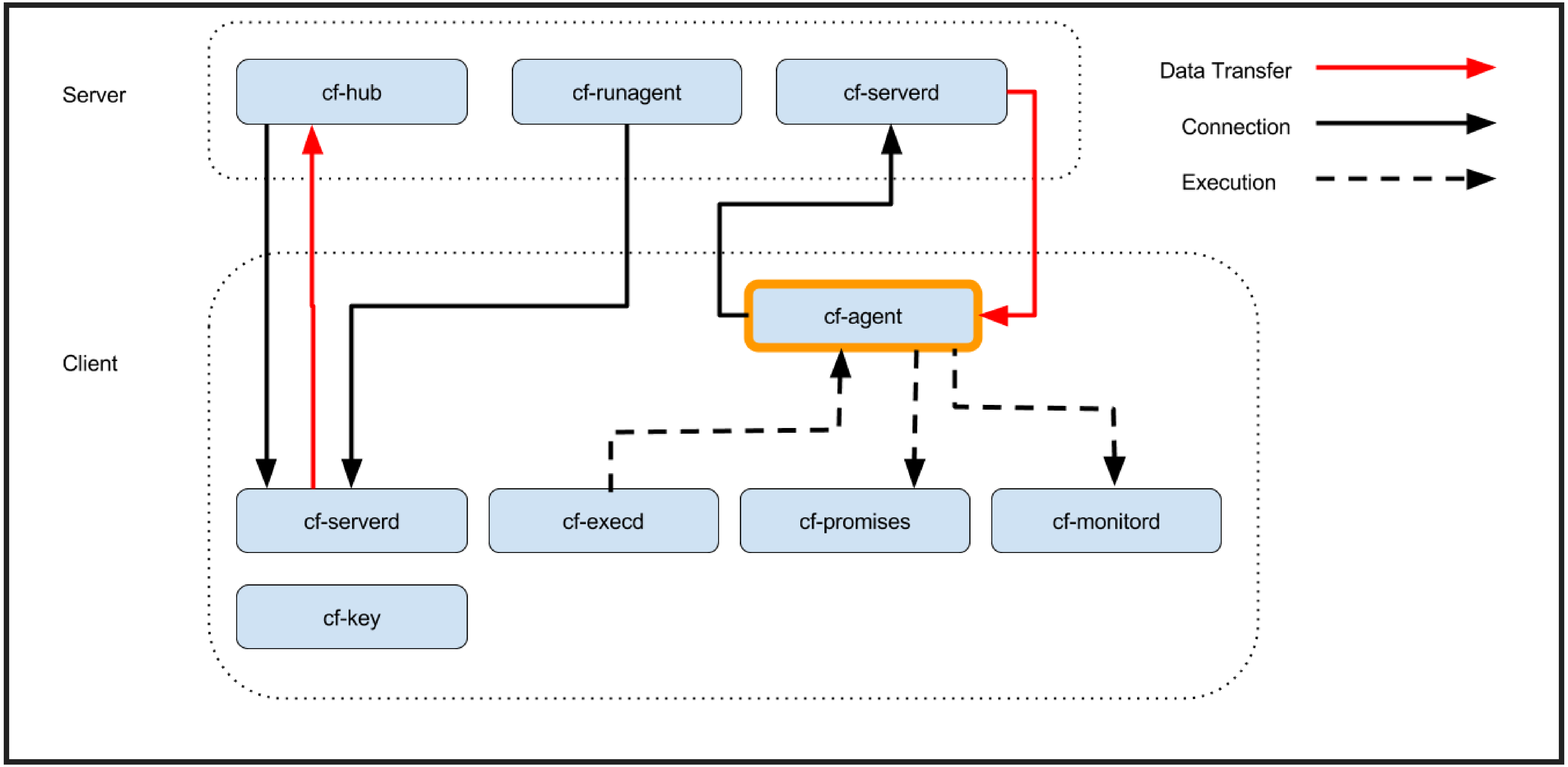
# CF-EXECD


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# `CF-EXECD`

- Executes `cf-agent` periodically
- Controls period and splay time
- Collects, stores and sends output
- Evaluates "`common`" bundles
- Obeys "`executor control`" body

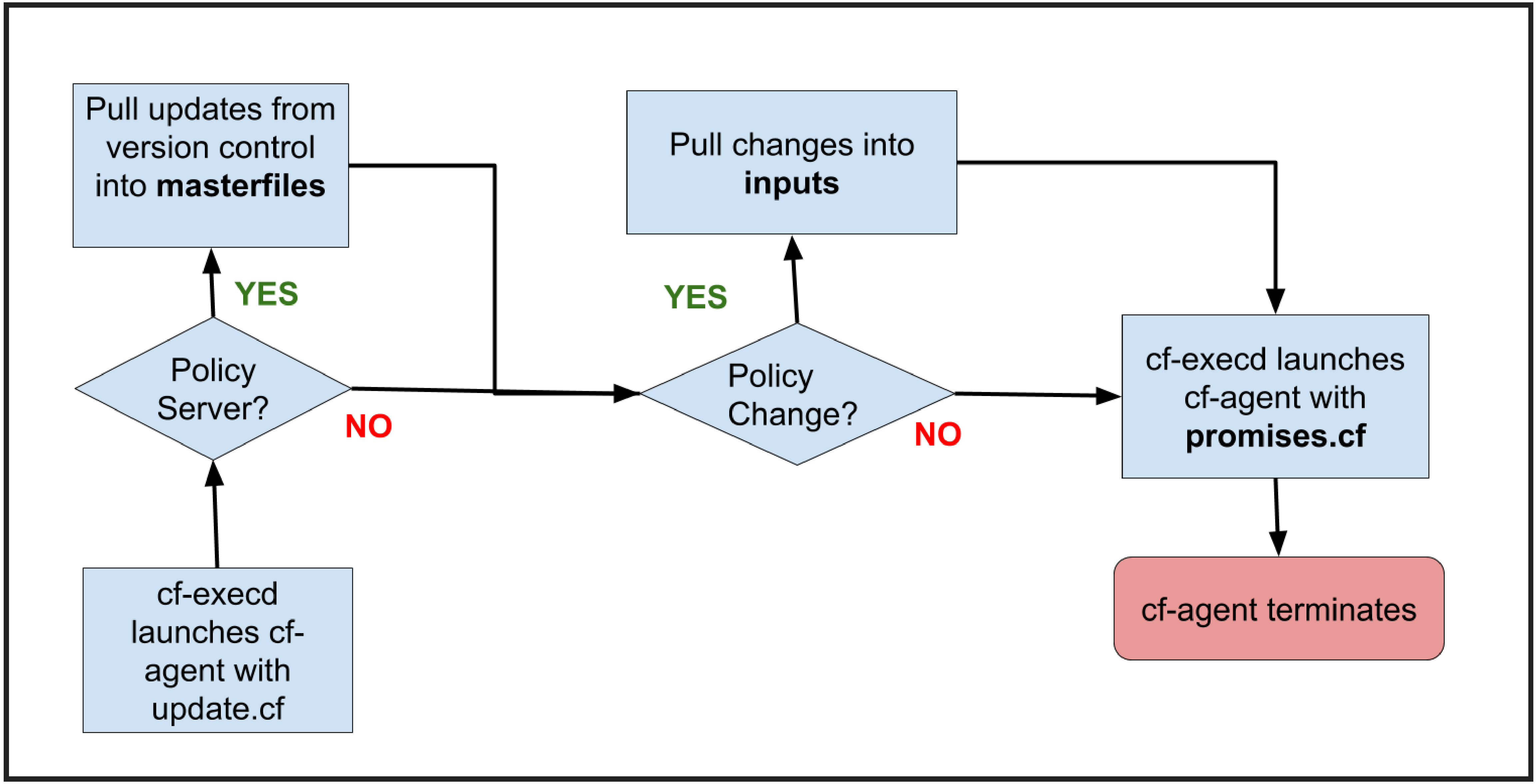
# CF-AGENT


Server
cf-hub
cf-runagent
cf-serverd
Data Transfer
Connection
Execution
Client
cf-agent
cf-serverd
cf-execd
cf-promises
cf-monitord
cf-key

# CF-AGENT

- The Catalyst or instigator of change
- Evaluates "`agent`" and "`common`" bundles
- Obeys "`common control`" and "`agent control`" bodies.
- By default runs `/var/cfengine/inputs/promises.cf`

# POLICY FLOW DIAGRAM



Pull updates from version control into **masterfiles**
Pull changes into **inputs**
YES
YES
Policy Server?
NO
Policy Change?
NO
cf-execd launches cf-agent with **promises.cf**
cf-execd launches cf-agent with update.cf
cf-agent terminates

## POP QUIZ

- What needs to happen before remote agents can get policy from the hub?
- Where is the policy that the agent runs by default?
- What port needs to be open bidirectionally?
- What is the meaning of life, the universe and everything?

# CLIENT/SERVER CHECKPOINT

- Any system can be a policy server
- Bootstrapping establishes trust relationship
- Policy server bootstraps to itself
- Agent runs locally cached policy in `/var/cfengine/inputs`
- Communicates over port **`5308`** (tcp)
- Agents downloads policy from server
- Hub downloads reports from remote agents

# LANGUAGE

- **Promise Theory** is a model of **voluntary cooperation** between individual autonomous actors.
- The fundamental underlying philosophy that CFEngine is based on.

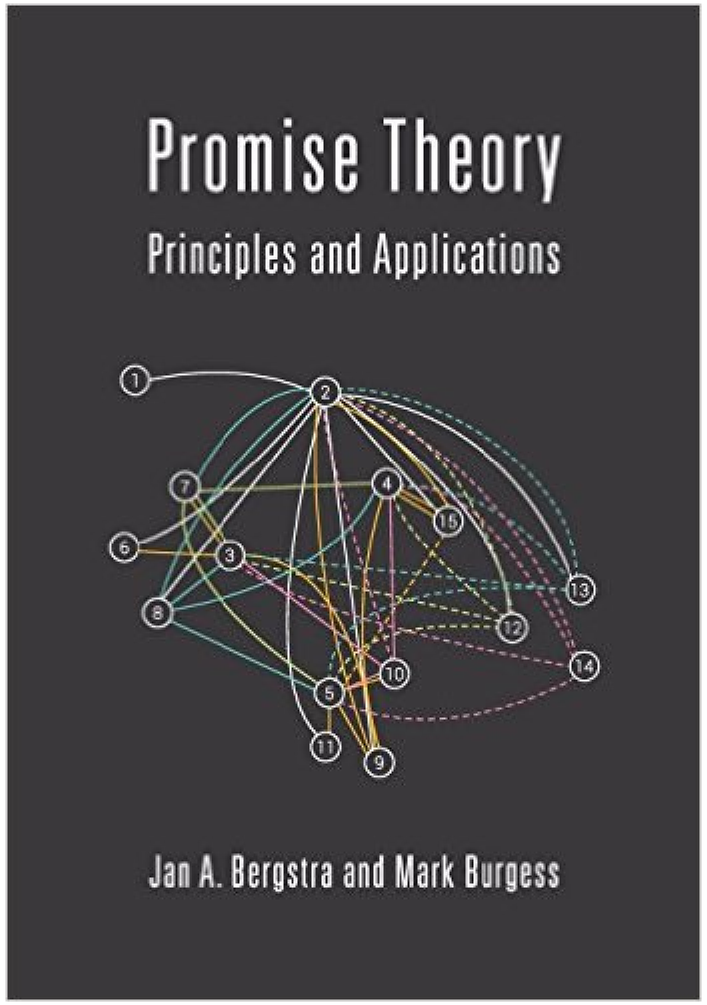


Figure 15: Promise Theory Primcipals and Applications

# PROMISES

- A promise is a **statement of intent**
    - The webserver process promises to be running
    - The file promises to have specific permissions
- An agent can only make promises about itself

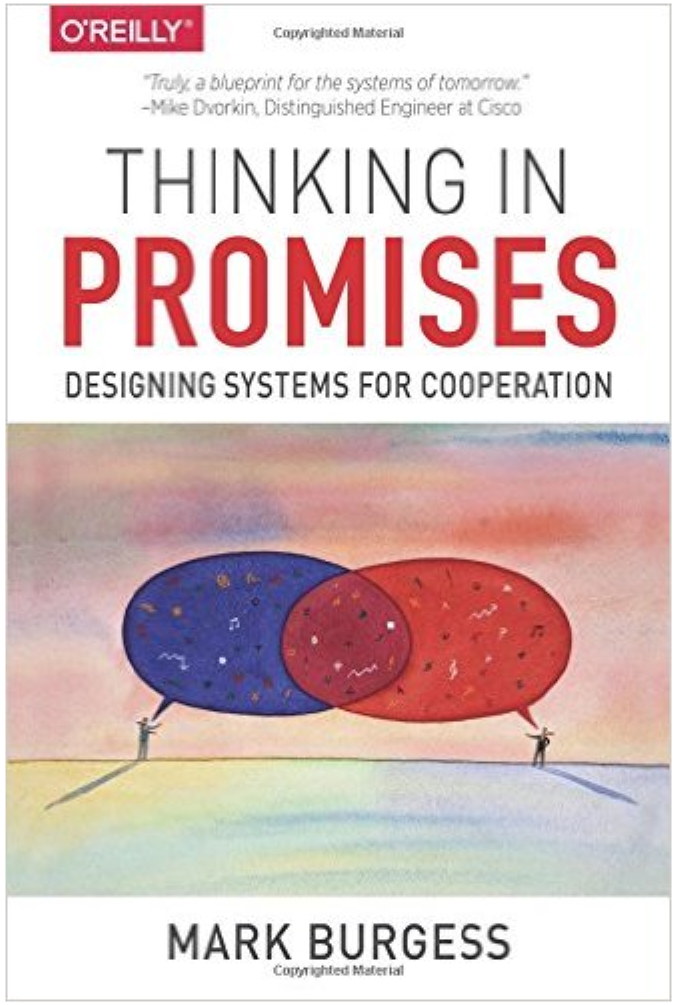

O'REILLY®
Copyrighted Material
"Truly, a blueprint for the systems of tomorrow."
–Mike Dvorkin, Distinguished Engineer at Cisco
THINKING IN
PROMISES
DESIGNING SYSTEMS FOR COOPERATION
MARK BURGESS
Copyrighted Material

Figure 16: Thinking in promises

# PROMISE OUTCOMES/RESULTS

- `kept`
- `repaired`
- `not_kept`
    - `failed`
    - `denied`
    - `timeout`

**Note:** Compound promises **can** have **multiple outcomes** at the same time (`not_kept` & `kept`).

# PROMISE TYPES

| | | |
|---|---|---|
| commands | files | services |
| packages | methods | defaults |
| meta | classes | reports |
| vars | databases | guest_environments |
| processes | storage | users |
| access | roles | measurements |

There are many type of promises. The promise type available depend on the type of bundle used.

# ANATOMY OF A PROMISE

```
bundle type name
{
  promise_type:

    context::

      "promiser" -> { "optional", "stakeholder" }
        attribute_1 => value_1,
        attribute_2 => value_2,
        attribute_3 => value_3,
        attribute_n => value_n;
}
```

# BUNDLES

- Collection of promises grouped logically
- Can take parameters
- Maintain variable state
- Are **not functions**

# EXAMPLE BUNDLES HOLD STATE

```
bundle agent main
{
  methods:
      "" usebundle => set_item_in_bag( "hammer", "slightly damaged");
      "" usebundle => set_item_in_bag( "pickaxe", "perfect");
      "" usebundle => set_item_in_bag( "flask", "half full");
      "" usebundle => report_bag_contents;

}
bundle agent set_item_in_bag(item, value)
{
  vars: "$(item)" string => "$(value)";
}
bundle agent report_bag_contents
{
  vars:
    "l" slist => variablesmatching("default:set_item_in_bag\..*");
    "a[$(l)]" string => nth( string_split( "$(l)", "\.", 2 ), 1);
  reports: "You look in the bag and see ...."; "A $($(l)) $(a[$(l)])";
}
```

```
R: You look in the bag and see ....
R: A half full flask
R: A slightly damaged hammer
R: A perfect pickaxe
```

# BUNDLES FOR LOGICAL ABSTRACTION

For example, a bundle to configure Apache might:

- install the `apache2` package
- edit the configuration file
- copy the web server content
- configure file-system permissions
- ensure the httpd process is running
- restart the httpd process when necessary

# BUNDLE TYPES

Bundles have a type which specify how they can be used.

`common`

evaluated by **all** components

`agent`

evaluated by `cf-agent`

`edit_line`

evaluated by `cf-agent` for files type promises

`edit_xml`

evaluated by `cf-agent` for files type promises

`server`

evaluated by `cf-serverd`

`monitor`

evaluated by `cf-monitord`

# NORMAL ORDER

- The order in which CFEngine executes **promise types**
  - Classes then Vars are evaluated during policy parsing and pre-eval.
  - Promises are evaluated from top to bottom respecting class restrictions
  - Bundles listed in the bundlesequence or activated via methods are given 3 pass evaluation.

# BODIES

- Collection of attributes
- Have a type ( e.g. `classes`, `perms`, `copy_from` )
- Can take parameters
- Can be inherited (3.8+)
- Cab be *defaulted* for a given promise *type* (3.9+)

There are **MANY** body types.

# POP QUIZ

- What is a bundle?
- What is the fundamental underlying philosophy that CFEngine is based on?
- When is the next break?
- Which component collects reports from remote agents?
- What types of bundles are there?

# MAKING DECISIONS: CONTEXT CLASSES AND EXPRESSIONS

## CLASSES

- Restrict promises based on context.
- Usable in bundles and bodies
- Used for making decisions
- Can be combined with expressions

# SHOW CLASSES

```
cf-promises --show-classes
```

```
| Class name                                                   Meta tags                  |
| 127_0_0_1                                                    inventory                  | attri
| 172_17_0_1                                                   inventory                  | attri
| 4_cpus                                                       source=agent               | deriv
| 64_bit                                                       source=agent               | hardc
| Day15                                                        time_based                 | sourc
| Evening                                                      time_based                 | sourc
| GMT_Day16                                                    time_based                 | sourc
| GMT_Hr03                                                     time_based                 | sourc
| GMT_Hr03_Q2                                                  time_based                 | sourc
| GMT_Hr3                                                      time_based                 | sourc
| GMT_Lcycle_0                                                 time_based                 | sourc
| GMT_Min15_20                                                 time_based                 | sourc
| GMT_Min16                                                    time_based                 | sourc
| GMT_Night                                                    time_based                 | sourc
| GMT_October                                                  time_based                 | sourc
| GMT_Q2                                                       time_based                 | sourc
| GMT_Sunday                                                   time_based                 | sourc
| GMT_Yr2016                                                   time_based                 | sourc
| Hr22                                                         time_based                 | sourc
| Hr22_Q2                                                      time_based                 | sourc
| Lcycle_0                                                     time_based                 | sourc
| Min15_20                                                     time_based                 | sourc
| Min16                                                        time_based                 | sourc
| October                                                      time_based                 | sourc
| PK_SHA_43c979e264924d0b4a2d3b568d71ab8c768ef63487670f2c51cd85e8cec63834 inventory       | attri
| Q2                                                           time_based                 | sourc
| Saturday                                                     time_based                 | sourc
| Yr2016                                                       time_based                 | sourc
| Yr2016                                                       time_based                | sourc
| any                                                          source=agent              | hardc
| cfengine                                                     inventory                 | attri
| cfengine_3                                                   inventory                 | attri
| cfengine_3_9                                                 inventory                 | attri
| cfengine_3_9_1                                               inventory                 | attri
| common                                                       cfe_internal              | sourc
| compiled_on_linux_gnu                                        source=agent              | hardc
| debian                                                       inventory                 | attri
| debian_stretch                                               inventory                 | attri
| enterprise                                                   inventory                 | attri
| enterprise_3                                                 inventory                 | attri
| enterprise_3_9                                               inventory                 | attri
| enterprise_3_9_1                                             inventory                 | attri
| enterprise_edition                                           inventory                 | attri
| feature                                                      source=agent              | hardc
| feature_curl                                                 source=agent              | hardc
| feature_def                                                  source=agent              | hardc
| feature_def_json                                             source=agent              | hardc
| feature_def_json_preparse                                    source=agent              | hardc
| feature_xml                                                  source=agent              | hardc
| feature_yaml                                                 source=agent              | hardc
| have_aptitude                                                inventory                 | attri
| ipv4_127                                                     inventory                 | attri
| ipv4_127_0                                                   inventory                 | attri
| ipv4_127_0_0                                                 inventory                 | attri
| ipv4_127_0_0_1                                               inventory                 | attri
| ipv4_172                                                     inventory                 | attri
| ipv4_172_17                                                  inventory                 | attri
| ipv4_172_17_0                                                inventory                 | attri
| ipv4_172_17_0_1                                              inventory                 | attri
| linux                                                        inventory                 | attri
| linux_4_4_0_43_generic                                       source=agent              | deriv
| linux_x86_64                                                 source=agent              | deriv
| linux_x86_64_4_4_0_43_generic                                source=agent              | deriv
| linux_x86_64_4_4_0_43_generic__63_Ubuntu_SMP_Wed_Oct_12_13_48_03_UTC_2016 source=agent | deriv
| linux_x86_64_4_4_0_45_generic__65_ubuntu_smp_wed_oct_12_13_48_03_utc_2016 source=agent | deri
| mac_02_42_2d_73_e9_95                                         inventory             | attri
| net_iface_docker0                                             source=agent          | hardc
| net_iface_lo                                                  source=agent          | hardc
| nickanderson_thinkpad_w550s                                   source=agent          | deriv
| nova                                                          inventory             | attri
| nova_3                                                        inventory             | attri
| nova_3_9                                                      inventory             | attri
| nova_3_9_1                                                    inventory             | attri
| nova_edition                                                  source=agent          | hardc
| systemd                                                       inventory             | attri
| ubuntu                                                        inventory             | attri
| ubuntu_16                                                     inventory             | attri
| ubuntu_16_4                                                   inventory             | attri
| x86_64                                                        source=agent          | deriv
```

# CLASS EXPRESSIONS

Table 1: Class Expressions

| | |
|---|---|
| **.** (dot) | **AND** |
| **&** (ampersand) | **AND** |
| **\|** (pipe) | **OR** |
| **!** (exclamation) | **NOT** |
| **()** (parenthesis) | **grouping** |

- It's more common to use `.` than **&** to express AND

# CLASSES EXAMPLE

```
bundle agent main
{
  files:
    linux.(Sunday|Saturday)::
      "/etc/nologin" -> { "Human Resources" }
        create => "true",
        comment => "Disallow non-root logins on the weekend.
                    We believe in work-life balance, and
                    encourage it.";

    linux.!(Sunday|Saturday)::
      "/etc/nologin" -> { "Business Operations" }
        delete => tidy,
        comment => "People need to be able to log in for them
                  to do their work during the week";
}
```

## CLASS TYPES

- Hard Classes
- Soft Classes

## HARD CLASSES

- Defined by agent
- Not configurable
- Always available
- Discovered each run
- Cannot be undefined

`redhat`, `Thursday`, `linux`

# SOFT CLASSES

- Defined by policy
- Based on anything
- Available after definition
- Can persist for period of time
- Can be namespace or bundle scoped
- Can be undefined

`webserver`, `prod_env`, `north_america`

# VALID CLASS NAMES

- Allowed characters include `[a-zA-Z0-9_]`
- **Canonify** a string to produce a valid class name (by converting invalid characters to _).
- Automatically canonified **when defined**

**Note:** Classes are **NOT** automatically canonified when **checked**.

# EXAMPLE OF AUTOMATIC CANONIFICATION

## examples/00-01-classes_canonification.cf

```
bundle agent main
{
  vars:
    "my_class_name" string => "Invalid-Class/Name!";
    "c_my_class_name" string => canonify( "$(my_class_name)" );

  classes:
    "$(my_class_name)" expression => "any";

  reports:

    "'$(my_class_name)' is **NOT** a class that is defined"
      unless => "$(my_class_name)";

    "'$(c_my_class_name)' **IS** a defined class"
      if => canonify( $(my_class_name) );
}
```

```
R: 'Invalid-Class/Name!' is **NOT** a class that is defined
R: 'Invalid_Class_Name_' **IS** a defined class
```

## CLASS SCOPE

- Namespace scoped classes are accessible from any bundle.
- Persist until end of agent run or explicitly undefined.
- Bundle scoped classes are only accessible from within the bundle the class was defined.
- All hard classes are namespace scoped

# CLASS SCOPE RULES

Table 2: Default scope for `classes` **type** promises

| Bundle Type | Scope |
|---|---|
| `common` | `namespace` |
| `agent` | `bundle` |

- `classes` bodies default to **namespace scope**

**Pro Tip:** Use **`bundle`** scoped classes whenever possible.

# DEFINE CUSTOM CLASSES BY EXPRESSION

## examples/00-10-classes_by_expression.cf

```
bundle agent main
{
  classes:
    "weekend" or => { "Saturday", "Sunday" };
    "weekday" not => { "weekend" };

    "business_hours"
      expression => "weekday.(Hr9|Hr10|Hr11|Hr13|Hr14|Hr15|Hr16|Hr17)",
      comment => "Weekdays from 9-5 excluding the lunch hour.";

    "webserver"
      expression => regcmp( "www.*", $(sys.fqhost) ),
      comment => "Identify webservers based on their name";

    "north_america"
      expression => iprange( "10.1.0.0/16" );
}
```

# DEFINE CUSTOM CLASSES BY PROMISE OUTCOME

## examples/00-10-classes_define_based_on_promise_outcome.cf

```
bundle agent main
{
  vars:
    "config[PermitRootLogin]" string => "no";

  files:
    "/etc/ssh/sshd_config"
      edit_line => set_line_based("$(this.bundle).config", " ", "\s+", ".*", "\s*#\s*"),
      classes => scoped_classes_generic("bundle", "sshd_config");

  services:
    sshd_config_repaired::
      "sshd"
        service_policy => "restart",
        comment => "For sshd to pick up changed config it must be restarted."
}
```

# EXAMPLE - TRADITIONAL CLASS EXPRESSIONS

`examples/00-10-classes_traditional_expression.cf`

```
bundle agent main
{
  reports:
    linux.!(Saturday|Sunday)::
      "This is a linux host";
      "Today is not Saturday or Sunday";
}
```

Traditional class expressions apply until the next class expression, promise type, or end of bundle. If unspecified promises are in the `any` context.

# EXAMPLE - IFVARCLASS

## `examples/00-10-classes_example_ifvarclass.cf`

```
bundle agent main
{
  vars:
    "platforms" slist => { "linux", "windows" };

  reports:
    "I am a $(platforms) host"
      ifvarclass => "$(platforms)";
}
```

- Restrict individual promises

# EXAMPLE - IF AND UNLESS

## Better readability with if/unless

examples/00-10-classes_example_if_and_unless.cf

```
bundle agent main
{
  vars:
    "platforms" slist => { "linux", "windows" };

  reports:
    "I am a $(platforms) host"
      if => "$(platforms)";

    "I was made by Microsoft"
      unless => "!windows";
}
```

# EXAMPLE - VARIABLE CLASS EXPRESSIONS

## More flexibility with variable class expressions

`examples/00-10-classes_example_variable_class_expressions.cf`

```
bundle agent main
{
  vars:
    "platforms" slist => { "linux", "windows" };

  reports:
    "$(platforms)"::
      "I am a $(platforms) host";
}
```

# POP QUIZ

- What types of classes are there?
- What is the difference?
- What are valid class characters?
- What is the default scope for classes defined as the result of a promise outcome using a classes body?

# DATA TYPES

# VARIABLE SCOPING

- `$(variable)`,`@(list)`
- Use fully qualified variable names when accessing variables in other bundles.
  - `$(bundle.variable)`,`@(bundle.list)`
  - `$(namespace:bundle.variable)`, `@(namespace:bundle.list)`

# EXAMPLE - STRINGS

examples/00-01-strings.cf

```
bundle agent main
{
  vars:
   "string1" string => "one";
    "string2" string => "strings
can be multi-line";
    "string3" string => "with \"quotes\"";
    "string4" string => 'or "quotes"';

  reports:
    "string1 = '$(string1)'";
    "string2 = '$(string2)'";
    "string3 = '$(string3)'";
    "string4 = '$(string4)'";
}
```

```
R: string1 = 'one'
R: string2 = 'strings
can be multi-line'
R: string3 = 'with "quotes"'
R: string4 = 'or "quotes"'
```

# EXAMPLE - NUMBERS

examples/00-02-numbers.cf

```
bundle agent main
{
  vars:
    "var1" int => "1";
    "var2" int => "10K";
    "var3" real => "1.2";
    "var4" real => "10e-5";
    "var5" int => "inf";

  reports:
    "var1 = '$(var1)'";
    "var2 = '$(var2)'";
    "var3 = '$(var3)'";
    "var4 = '$(var4)'";
    "inf = '$(var5)'";
}
```

```
R: var1 = '1'
R: var2 = '10240'
R: var3 = '1.200000'
R: var4 = '0.000100'
R: inf = '999999999'
```

# EXAMPLE - LISTS

## examples/00-04-lists.cf

```
bundle agent main
{
  vars:
    "var1" ilist => { 1, 2, "3", "4" };
    "var2" rlist => { "1.2", "2.0", "3.3" };
    "var3" slist => { "one", "two", three,
                      @(var1), @(var2),
                    };
    "var4" real => sum( var2 );

  reports:
    "var1 = '$(var1)'";
    "var2 = '$(var2)'";
    "var3 = '$(var3)'";
    "var4 = '$(var4)'";
}
```

```
R: var1 = '1'
R: var1 = '2'
R: var1 = '3'
R: var1 = '4'
R: var2 = '1.2'
R: var2 = '2.0'
R: var2 = '3.3'
R: var3 = 'one'
R: var3 = 'two'
R: var3 = 'three'
R: var3 = '1'
R: var3 = '2'
R: var3 = '3'
R: var3 = '4'
R: var3 = '1.2'
R: var3 = '2.0'
R: var3 = '3.3'
R: var4 = '6.500000'
```

# EXAMPLE - LIST ITERATION

## examples/00-03-list_iteration.cf

```
bundle agent main
{
  vars:
    "numbers" slist => { "1", "2", "3" };
    "colors" slist => { "red", "green", "blue" };

  reports:
    "$(numbers)";
    "$(colors)";
    "$(numbers) with $(colors)";
    "$(colors) with $(numbers)";
}
```

```
R: 1
R: 2
R: 3
R: red
R: green
R: blue
R: 1 with red
R: 2 with red
R: 3 with red
R: 1 with green
R: 2 with green
R: 3 with green
R: 1 with blue
R: 2 with blue
R: 3 with blue
R: red with 1
R: green with 1
R: blue with 1
R: red with 2
R: green with 2
R: blue with 2
R: red with 3
R: green with 3
R: blue with 3
```

# EXAMPLE - "CLASSIC" ARRAYS

## examples/00-05-classic_arrays.cf

```
bundle agent main
{
  vars:
    "file[motd]"  string => "/etc/motd";
    "file[fstab]" string => "/etc/fstab";

    "file_idx" slist => getindices( file );
    "files" slist => getvalues( file );

  reports:
   "The key '$(file_idx)' has the value '$(file[$(file_idx)])'";
   "file: '$(files)'";
}
```

```
R: The key 'fstab' has the value '/etc/fstab'
R: The key 'motd' has the value '/etc/motd'
R: file: '/etc/fstab'
R: file: '/etc/motd'
```

# EXAMPLE - DATA CONTAINERS

## examples/00-06-data_containers.cf

```
bundle agent main
{
  vars:
    "server" string => "mirror.int.cfengine.com";
    "repos"
      data => parsejson('{
                "rhel6_updates": {
                  "id": { "value":"RHEL6_UPDATES" },
                  "name": { "value": "RHEL 6.x Updates" },
                  "baseurl": { "value": "https://$(server)/RHEL6/updates" }
                }
              }');

    "idx" slist => getindices( repos );

  reports:
    "URL = '$(repos[$(idx)][baseurl][value])'";
}
```

```
R: URL = 'https://mirror.int.cfengine.com/RHEL6/updates'
```

## POP QUIZ

- Which component makes changes to your system?
- Which component schedules agent runs?
- What symbol terminates a promise?
- What is your quest?
- What are promise comments used for?
- What types of variables are available?
- How are decisions made?

# GIT PRIMER

# GIT

Git is the most popular modern version control management tool. Github, Bitbucket, and GitLab all provide great hosted and on prem repository management solutions.

Using a git management system is reccomended for implementing access controls and improved collaboration with regard to policy and systems management.

# GETTING STARTED

## Log into the policy server

```
[user@workstation] $ vagrant ssh hub
[vagrant@hub] $ sudo -i
```

## Configure git author

```
[root@hub masterfiles] # git config -–global user.name “Mr. Slate”
[root@hub masterfiles] # git config -–global user.email “bossman@slateco.com”
[root@hub masterfiles] # git config --global push.default simple
```

# CLONE BUILTIN REPOSITORY

```
[root@hub masterfiles] # git clone /opt/cfengine/masterfiles.git /vagrant/masterfiles
Cloning into '/vagrant/masterfiles'...
```

# ADD A FILE TO THE REPOSITORY

```
[root@hub masterfiles] # cd /vagrant/masterfiles
[root@hub masterfiles] # ls
cfe_internal  Changelog  controls  def.cf  lessons  lib  libraries  promises.cf  services  sketc
[root@hub masterfiles] # echo hi > file
[root@hub masterfiles] # git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file
nothing added to commit but untracked files present (use "git add" to track)
```

# STAGE AND COMMIT THE CHANGES

```
[root@hub masterfiles] # git add file
[root@hub masterfiles] # git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   file
#
# git commit -m “Testing git workflow”
[master c886caf] Testing git workflow
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file
```

# PUBLISH THE CHANGE

```
[root@hub masterfiles] # git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
nothing to commit (working directory clean)
[root@hub masterfiles] # git push
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /opt/cfengine/masterfiles.git
   ee31801..94b8151  master -> master
# git status
# On branch master
nothing to commit (working directory clean)
```

# MODIFY A FILE

```
[root@hub masterfiles] # echo HI >> file
[root@hub masterfiles] # git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   file
#
no changes added to commit (use "git add" and/or "git commit -a")
```

# DIFF TO VALIDATE DETAIL OF CONTENT CHANGE

```
[root@hub masterfiles] # git diff
diff --git a/file b/file
index 45b983b..313352b 100644
--- a/file
+++ b/file
@@ -1 +1,2 @@
 hi
+HI
```

# STAGE AND COMMIT THE CHANGES

```
[root@hub masterfiles] # git add file
[root@hub masterfiles] # git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   file
#
# git commit -m "Modified file"
[master fd94885] Modified file
 1 files changed, 1 insertions(+), 0 deletions(-)
# git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commits.
#
nothing to commit (working directory clean)
```

# PUBLISH THE CHANGE

```
[root@hub masterfiles] # git push
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 493 bytes, done.
Total 5 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (5/5), done.
To /opt/cfengine/masterfiles.git
   c886caf..fd94885  master -> master
```

# REMOVE A FILE

```
[root@hub masterfiles] # git rm file
# rm 'file'
[root@hub masterfiles] # git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    file
#
# git commit -m "Remove file"
[master 4c9d46d] Remove file
 1 files changed, 0 insertions(+), 3 deletions(-)
 delete mode 100644 file
```

# PUBLISH THE CHANGE

```
[root@hub masterfiles] # git push
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 230 bytes, done.
Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
To /opt/cfengine/masterfiles.git
   92660a9..4c9d46d  master -> master
```

# PLACE MASTERFILES INTO THE REPOSITORY

```
[root@hub masterfiles] # rsync -avz /var/cfengine/masterfiles/ .
[root@hub masterfiles] # git add -A
[root@hub masterfiles] # git commit -m "Seed repository with masterfiles"
[root@hub masterfiles] # git push
```

# CONFIGURE MASTERFILES FOR GIT INTEGRATION

```
[root@hub masterfiles]# echo '{ "classes": { "cfengine_internal_masterfiles_update": [ "policy_s
[root@hub masterfiles]# git add def.json
[root@hub masterfiles]# git commit -m "Enable automatic masterfiles update from repository"
[master c206654] Enable automatic masterfiles update from repository
 1 file changed, 1 insertion(+)
 create mode 100644 def.json
[root@hub masterfiles]# git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 370 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /opt/cfengine/masterfiles.git
   a181449..c206654  master -> master
```

# ACTIVATE CURRENT UPDATE POLICY WITH ENABLEMENT CLASS

```
[root@hub masterfiles]# cf-agent --no-lock \
--verbose \
--define cfengine_internal_masterfiles_update \
--file update.cf
```

# VERIFY PRESENCE OF DEF.JSON

```
[root@hub masterfiles]# cat /var/cfengine/masterfiles/def.json
```

# REVIEW

Now when the agent runs on hosts that define `policy_server` policy will be activated that keeps `/var/cfengine/masterfiles` up to date with whats in the repository as configured in mission portal (which by default is `/opt/cfengine/masterfiles.git`).

This means new policy will be distributed as you push it into the repository. Continuous delivery FTW!

# GIT COMMAND REFERENCE

| | |
|---|---|
| `git status` | `git pull --rebase` |
| `git diff` | `git grep` |
| `git add` | `git log` |
| `git diff --cached` | `git checkout -b new_feature` |
| `git commit` | `git push origin <branch>` |

# POP QUIZ

- Who was Mr. Slate?
- How do you get an overview of the changes to your clone?
- How can you easily search a git repository?

# EXAMPLES AND EXERCISES

# ENABLE SERVICESAUTORUN

```
[root@hub masterfiles]# cat > def.json <<EOF
{
    "classes": {
        "cfengine_internal_masterfiles_update": [
            "policy_server"
        ],
        "services_autorun": [ "any" ]
    }
}
EOF
```

# INSPECT

```
[root@hub masterfiles]# git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   def.json
#
no changes added to commit (use "git add" and/or "git commit -a")
```

# WHAT YOU EXPECT

```
[root@hub masterfiles]# git diff
diff --git a/def.json b/def.json
index 0b1c721..bf3b515 100644
--- a/def.json
+++ b/def.json
@@ -1 +1,8 @@
-{ "classes": { "cfengine_internal_masterfiles_update": [ "policy_server" ] } }
+{
+    "classes": {
+        "cfengine_internal_masterfiles_update": [
+            "policy_server"
+        ],
+        "services_autorun": [ "any" ]
+    }
+}
```

# PUBLISH THE CHANGE

```
[root@hub masterfiles]# git add def.json
[root@hub masterfiles]# git commit -m "Enable autorun"
[master 7e57301] Enable autorun
 1 file changed, 8 insertions(+), 1 deletion(-)
[root@hub masterfiles]# git push origin master
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /opt/cfengine/masterfiles.git
   c206654..7e57301  master -> master
```

# EXAMPLE - CREATE A FILE

Copy this example to `services/autorun/nologin.cf`

`examples/00-20-example-create_file.cf`

```
bundle agent no_login
{
  meta:
    "tags" slist => { "autorun" };

  files:
      "/etc/nologin"
        create => "true";
}
```

# EXAMPLE - UPDATE FILE

## examples/00-20-example-update_file.cf

```
bundle agent cf_agent_heartbeat
{
  meta:
    "tags" slist => { "autorun" };

  files:
    "$(sys.statedir)/$(this.bundle)" -> { "Monitoring" }
      create  => "true",
      touch   => "true",
      classes => scoped_classes_generic("bundle", "heartbeat"),
      handle  => "cf_agent_heartbeat_thump",
      comment => "This policy simply updates the timestamp
                  of the file so an external monitoring
                  system can validate the system is working.";

  reports:
    DEBUG|DEBUG_cf_agent_heartbeat::

      "DEBUG $(this.bundle): Heartbeat"
        if => "heartbeat_repaired";

      "DEBUG $(this.bundle): Flatline"
        unless => "heartbeat_repaired";
}
```

# EXERCISE - CREATE A FILE

Write a policy that ensures `/etc/cron.allow` is present so that only users listed in the file are allowed to schedule jobs.

Test with `crontab -e`

# EXAMPLE - INSTALL A PACKAGE AND START A SERVICE

## examples/00-20-example-package_and_service.cf

```
bundle agent training_simple_website
{
  meta:
    host001::
      "tags" slist => { "autorun" };

  packages:
      "httpd"
        policy => "present",
        package_module => yum;

  services:
      "httpd"
        service_policy => "start";

  reports:
    DEBUG|DEBUG_training_simple_website::
      "DEBUG $(this.bundle)";
}
```

# EXAMPLE - CLASSIFY ROLE BASED ON HOSTNAME

## examples/00-20-example-classes-role_by_hostname.cf

```
bundle common classes_role_by_hostname
{
  classes:
    "env_prod"
      meta => { "inventory", "attribute_name=Role Classes" };
      or => {
              regcmp( "hub", $(sys.fqhost) ),
              regcmp( "^prd.*", $(sys.fqhost) ),
            };

    "webserver"
      expression => "host001",
      meta => { "inventory", "attribute_name=Role Classes" };

    "webserver"
      expression => regcmp( "^www.*", $(sys.fqhost) ),
      meta => { "inventory", "attribute_name=Role Classes" };
}
```

# EXAMPLE - CLASSIFY GEOGRAPHIC LOCATION BY NETWORK

## examples/00-20-example-classes-geographic_location_by_network.cf

```
bundle common classify_geo_location_by_network
{
  meta:
    "tags" slist => { "autorun" };

  vars:
    "nadc01_subnets"
      slist => { "172.16.25.0/25", "172.17.0.0/16",
                 "192.168.33.2/32",
               };

    "nadc02_subnets"
      slist => { "172.17.5.0/25", "172.42.0.0/16",
                 "192.168.33.3/32",
               };

    "sadc01_subnets"
      slist => { "172.19.4.0/23", "192.168.33.4-5" };

    "continent"
      string => ifelse("north_america", "North America",
                       "south_america", "South America",
                       "Unknown"),
      meta => { "inventory", "attribute_name=Continent" };
```

## Continued examples/00-20-example-classes-geographic_location_by_network.cf

```
  classes:
    "north_america"
      or => { "nadc01", "nadc02" };

    "south_america"
      expression => classmatch("^sadc\d+");

    "nadc01"
      expression => iprange( $(nadc01_subnets) );

    "nadc02"
      expression => iprange( $(nadc02_subnets) );

    "sadc01"
      expression => iprange( $(sadc01_subnets) );

  reports:
    "DEBUG|DEBUG_$(this.bundle)"::
      "DEBUG $(this.bundle): Continent = '$(continent)'";
}
```

# EXAMPLE - MANAGE KEY VALUE ENTRIES IN A CONFIG FILE

## manage_ssh in examples/00-20-example-key_value_config.cf

```
bundle agent manage_ssh
{
  meta:
    "tags" slist => { "autorun" };

  methods:
    "SSH Config Data"
      usebundle => ssh_config;

    "Manage SSH Config"
      usebundle => ssh_config_manage_kv("ssh_config.data");
}
```

# ssh_config in examples/00-20-example-key_value_config.cf

```
bundle agent ssh_config
{
  vars:
      "data[PermitRootLogin]" string => "no";
      "data[Protocol]" string => "2";
      "data[Port]" string => "22";

    DEBUG|DEBUG_ssh_config::
      "keys" slist => getindices(data);

  reports:
    DEBUG|DEBUG_ssh_config::
      "DEBUG $(this.bundle): $(keys) = '$(data[$(keys)])'";
}
```

# ssh_config_manage_kv in examples/00-20-example-key_value_config.cf

```
bundle agent ssh_config_manage_kv(data)
{
  vars:
    "config" string => "/etc/ssh/sshd_config";

  files:
    "$(config)"
      handle => "ssh_config_manage_kv_entries",
      edit_line => set_line_based("$(data)", " ", "\s+", ".*", "\s*#\s*"),
      classes => scoped_classes_generic("bundle", "sshd_config");

  classes:
    sshd_config_repaired::
      "config_valid"
        expression => returnszero("/usr/sbin/sshd -t -f $(config)", noshell),
        comment => "It's important that we don't restart the service with a
                    broken config, or the service will be down.";

  commands:
    sshd_config_repaired.config_valid::
      "$(paths.service)"
        handle => "ssh_config_manage_kv_restart_after_config_change",
        args => "sshd restart",
        comment => "The service must be restarted in order to pick up new
                    configuration settings.";

  reports:
      "DEBUG $(this.bundle): Reparied configuration"
        if => "sshd_config_repaired";
      "DEBUG $(this.bundle): Configuration Valid"
        if => "config_valid";

      "DEBUG $(this.bundle): Restarted sshd after config change"
        depends_on => { "ssh_config_manage_kv_restart_after_config_change" };
}
```

# EXAMPLE - MANAGE KEY VALUE ENTRIES OUTPUT

```
[root@hub masterfiles]# cf-agent -KID DEBUG_ssh_config,DEBUG_ssh_config_manage_kv
    info: Installing cfe_internal_non_existing_package...
R: DEBUG ssh_config: PermitRootLogin = 'no'
R: DEBUG ssh_config: Protocol = '2'
R: DEBUG ssh_config: Port = '22'
    info: Edit file '/etc/ssh/sshd_config'
R: DEBUG ssh_config_manage_kv: Reparied configuration
    info: Executing 'no timeout' ... '/sbin/service sshd restart'
  notice: Q: "...in/service sshd": Stopping sshd:          [  OK  ]
Q: "...in/service sshd": Starting sshd:                    [  OK  ]
    info: Last 2 quoted lines were generated by promiser '/sbin/service sshd restart'
    info: Completed execution of '/sbin/service sshd restart'
R: DEBUG ssh_config_manage_kv: Configuration Valid
R: DEBUG ssh_config_manage_kv: Restarted sshd after config change
```

# EXAMPLE - TEMPLATING A FILE

## examples/00-20-example-mustache_template_vars.cf

```
bundle agent motd
{
  meta:
    "tags" slist => { "autorun" };

  vars:
    "owner" string => "Bruce Wayne";

  files:
    "/etc/motd"
      template_method => "mustache",
      edit_template => "$(this.promise_filename).mustache";
}
```

# EXAMPLE - MUSTACHE TEMPLATE

## examples/00-20-example-mustachetemplatevars.cf.mustache

```
Welcome to {{{vars.sys.fqhost}}}!

For support issues please contact {{{vars.motd.owner}}}.
```

## TODO MUSTACHE PRO TIP: `-TOP-`

The `-top-` key is very useful for completely generic data models and debugging.

```
{{{%-top-}}}
```

## EXERCISE - EDIT A FILE

Write a policy that populates `/etc/issue` with legaleze.

## EXAMPLE - ENABLE SSHD BANNER

Write a policy that enables the ssh banner and directs it to `/etc/issue`. sshd should be restarted after changing its configuration.

# EXAMPLE - MULTIPLE OUTCOMES

## Setup for the example

```
sudo touch /tmp/immutable
sudo chatter +i /tmp/immutable
```

## examples/00-20-example-multiple_outcomes.cf

```
body file control { inputs => { "$(sys.libdir)/stdlib.cf" }; }
bundle agent main
{
  files:
    "/tmp/immutable"
      create => "true",
      edit_line =>  example_edit_line,
      classes => results("bundle", "my_id");

  vars:
    "classes" slist => classesmatching(".*my_id.*");

  reports:
    "Found Class = '$(classes)'";
}

bundle edit_line example_edit_line
{
  insert_lines:
    "I want to edit an immutable file";
}
```

```
   error: Can't copy file permissions from '/tmp/immutable' to '/tmp/immutable.cf-after-edit' -
   error: Unable to save file '/tmp/immutable' after editing
R: Found Class = 'my_id_failed'
R: Found Class = 'my_id_not_kept'
R: Found Class = 'my_id_error'
R: Found Class = 'my_id_kept'
R: Found Class = 'my_id_reached'
```

Reference the implementation of the `results` classes body in the stdlib.

## Cleanup immutable file

```
sudo chattr -i /tmp/immutable
sudo rm /tmp/immutable
```

# EXERCISE - REPORT

Write a policy that defines a name in separate parts (at least 3). Then have CFEngine report the name in a random order.

## Example Solution

```
bundle agent main
{
  vars:
    "name_parts" slist => { "Ronald", "Mck", "Donald" };
    "shuffled" slist => shuffle( name_parts, randomint(0, inf) );
    "name" string => join(" ", shuffled );

  reports:
    "$(name)";
}
```

```
R: Donald Mck Ronald
R: Donald Ronald Mck
R: Ronald Donald Mck
```

## EXCERCISE - TRIGGER AN ACTION WHEN A FILE CHANGES

Write policy to monitor a file for change. When a change is seen report `Winner Winner Chicken Dinner`. Manually edit the file, show how cfengine detects and reports on the change.

# EXERCISE - AUTOMATICALLY ABORT BASED ON FILE PRESENCE

Write a policy that will abort cfengine execution if the file `$(sys.statedir)/abort_agent_execution` exists.

- See abortclasses in the cfengine documentation

## EXERCISE - DELETE FILE BASED ON AGE

Write a policy that will delete `$(sys.statedir)/abort_agent_execution` if it is older than 1 hour.

# EXERCISE - KILL A PROCESS

Write a policy to kill irssi running on webservers. No need for our webservers to be connected to IRC.

```
cp /bin/sleep /tmp/irssi
/tmp/irssi 5000
```

# MPF & STDLIB

The "Default Masterfiles"

# OVERVIEW

**`promises.cf`**

The main entry. This is the first file the agent reads by default. This is the stem cell for the rest of your policy.

**`update.cf`**

This is a seperate **standalone** policy to manage updating policy and the cfengine agent itself.

# USER ENTRIES

- def.json
- services/main.cf
- services/autorun/*.cf
- promises.cf
- update.cf

# USEFUL TOOLS AND TIPS

# CF-LOCATE

**`cf-locate`**
: Command line tool to help locate and optionally display a body or bundle within a policy

# DEBUG REPORTS

Use standardized DEBUG reports for policy development and troubleshooting.

```
bundle agent main
{
  reports:
    DEBUG|DEBUG_this_bundle_name::
      "DEBUG $(this.bundle): ....";
    "DEBUG|DEBUG_$(this.bundle)"::
      "DEBUG $(this.bundle): ....";
 }
```

# PROMISE COMMENTS

Use promise comments to document **WHY** the promise is important.

```
bundle agent main
{
  services:
    "firewalld"
      policy => "start",
      comment => "If this service isnt running, then we have unnecessary
                  exposure and increase our risk of a security breach.";
}
```

# GETTING THE MOST FROM THE DOCUMENTATION

# ADDITIONAL RESOURCES

**VSA Training Material**

Self paced in depth tutorial based on Vertical Sysadmin training coursework

**Example Policy Layout**

An example policy layout

**CFEngine Spacemacs Layer**

The best editor is neither Emacs nor Vim, it's Emacs and Vim!

**`cf-locate`**

Find and optionally display body and bundle definitions within a policy set

**`cf-keycrypt`**

Tool to encrypt data with CFEngine public keys

- Allows to encrypt data for individual hosts using the public key
- Can be used with non host keys for "group" encryption

**`cf-profile`**

Displays time summaries from agent runs. Helps to find the bundles you spend the most time in.

**`vim_cf3`**

CFEngine 3 vim plugin with Syntax highlighting

**`reindent.pl`**

Re-indent CFEngine policy using this script that leverages the excellent cfengine3 mode in Emacs

**Sublime Text 3 CFEngine Beautifier**

Automatically reformat CFEngine policy in Sublime Text

**Sublime Text 3 Syntax Highlighter & Snippets**

CFEngine Syntax highlighting and snippets for Sublime Text

**(no term)**

`cfe-rsplaytime` Compute splaytime for a given host

**`cfe-profiler`**

Measures bundle execution time helping to uncover the most time consuming bundles.

# MORE

# DEFAULT BODIES

- Default Bodies in language concepts

3.9 introduced the ability to define a body that is used by all occurances of a given promise type unless otherwise specified.

For example, to set all file type promises to warn you can add the `files_action` body in the `bodydefault` namespace.

```
bundle agent main
{
  files:
    "/tmp/show_default_action"
      create => "true";

    "/tmp/show_explicit_action"
      create => "true",
      action => fix;
}

body action fix
{
  action_policy => "fix";
}

body file control
{
               "bodydefault"
  namespace => "bodydefault";

}
body action files_action
{
  action_policy => "warn";
}
```

```
warning: Warning promised, need to create file '/tmp/show_default_action'
```

# FEATURE MACRO

- Feature Macro documentation

You can conditionally parse policy based on compiled in features using this macro.

```
bundle agent main
{

@if feature(xml)
# the yaml library may not be compiled in
  vars: "container" data => parseyaml(...);
@endif

}
```

# SPECIAL TOPICS

# INTERNAL AGENT UPGRADE MECHANISM

- Place agent packages in `/var/cfengine/master_software_updates/$(sys.flavour)_s`
- Define `trigger_upgrade` for set of hosts. Ref `update_def.cf`

```
classes:
  "trigger_upgrade" or => { "host1", "host2", "redhat_5" };
```