THE UNIVERSITY *of* EDINBURGH
informatics

ConfSolve:
System Configuration with CSPs

John Hewson & Paul Anderson
University of Edinburgh

2012 Oxford Configuration Workshop - 13th January 2012

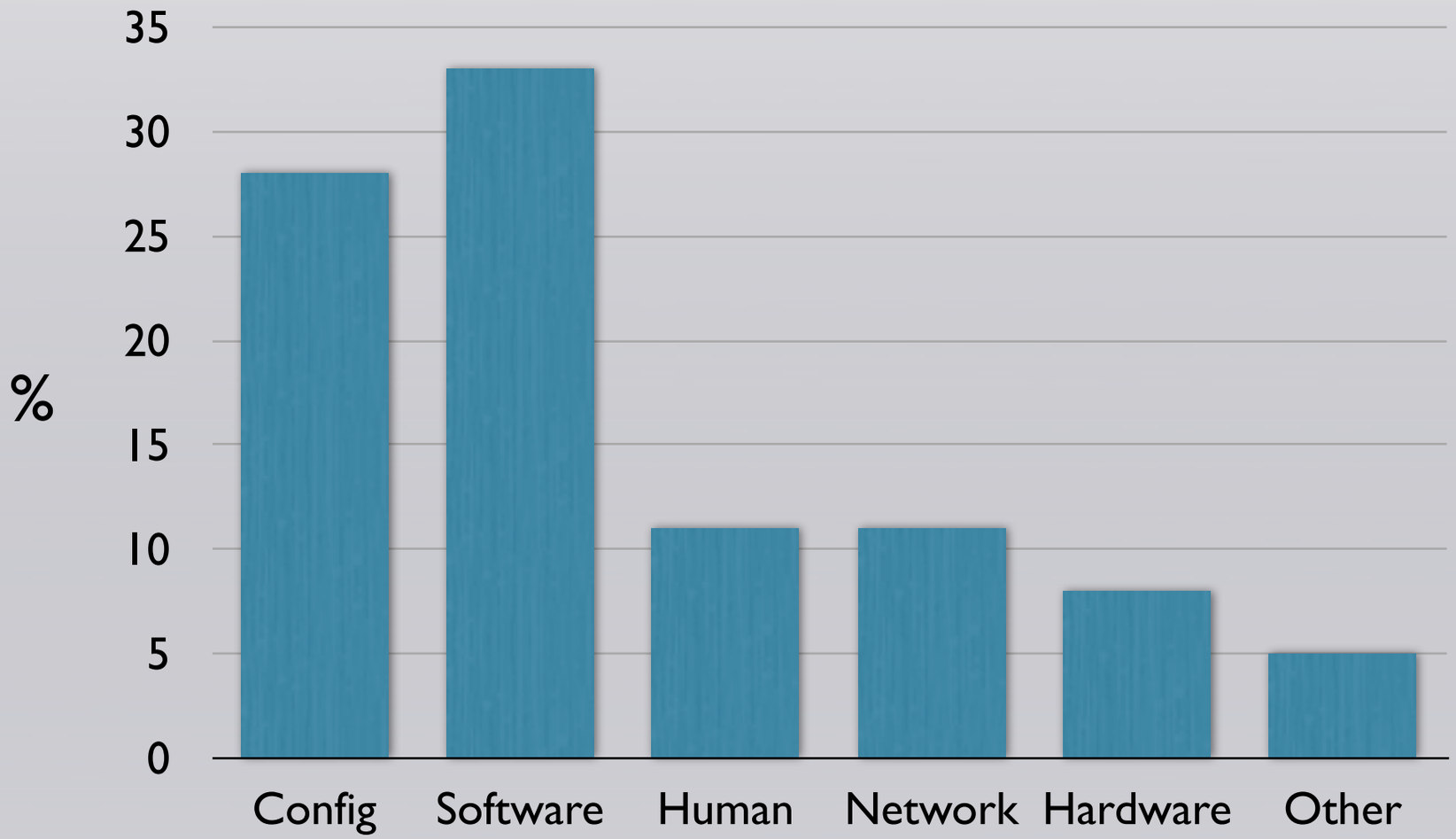
System Configuration

- physical machines, firewalls, networks, data-centres, clouds.

- Security: proving some invariants hold over both manually and automatically generated configurations.
- Scale of the cloud forces more automation
- but, bespoke enterprise systems are often more complex.

Google

Service disruption events by most likely cause at one of Google's main services, over 6 weeks (2009)



The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Hoelzle & Barroso, 2009.

Declarative Configuration

- LCFG - Anderson, 1993 - University of Edinburgh
- Cfengine - Burgess, 1993 - University College Oslo
- Bcfg2 - Desai, 2004 - Argonne National Laboratory

- Puppet - Reductive Labs, 2005

Declarative Configuration

```
package {'apache':
    ensure => installed
}
```

instead of

```
sudo apt-get –y install apache
```

What's Missing?

- The ability to verify that a configuration conforms to a model
- The ability to **infer** valid configurations from a model

THE UNIVERSITY of EDINBURGH
informatics

ConfSolve

- designed to be high-level and more familiar to system administrators:
- object oriented (like Puppet, CIM)
- inheritance
- primitives: integer, booleans, sets, enums
- objects, object references, sets of object references
- quantification and summation over decision variables

ConfSolve - Architecture

THE UNIVERSITY of EDINBURGH informatics

ConfSolve

3rd party

Example

```
enum Network { Public, Private }

class Machine {
  var cpu as int
  var memory as int
  var disk as int
  var network as Network

  where cpu == 16          // 16 * 1/2 CPU
  where memory == 16384    // 16 GB
  where disk == 2048       // 2 TB
  where network == Network.Public
}

class Role {
  var host as ref Machine
  var disk as int
  var cpu as int
  var memory as int
  var network as Network
}
```

Example (ctd.)

```
class SmallRole extends Role {
  where cpu == 1
  where memory == 768
  where disk <= 20
}

class LargeRole extends Role {
  where cpu == 4
  where memory == 3584
  where disk <= 490
}
```

Example (ctd.)

```
var machines as Machine[2]

var sql_server as LargeRole
where sql_server.disk == 412

var web_server as SmallRole
where web_server.disk == 15
where web_server.network == Network.Public
```

Example (ctd.)

```
var roles as ref Role[2]

where foreach (m in machines) {
  sum (r in roles where r.host == m) {
    r.cpu
  } <= m.cpu

  sum (r in roles where r.host == m) {
    r.memory
  } <= m.memory

  sum (r in roles where r.host == m) {
    r.disk
  } <= m.disk
}
```

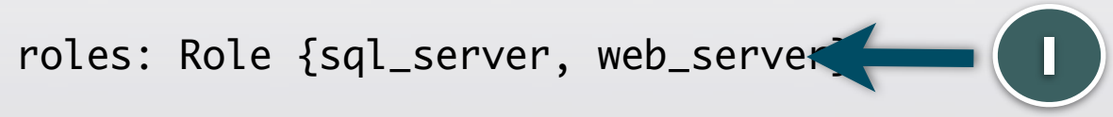
1

2

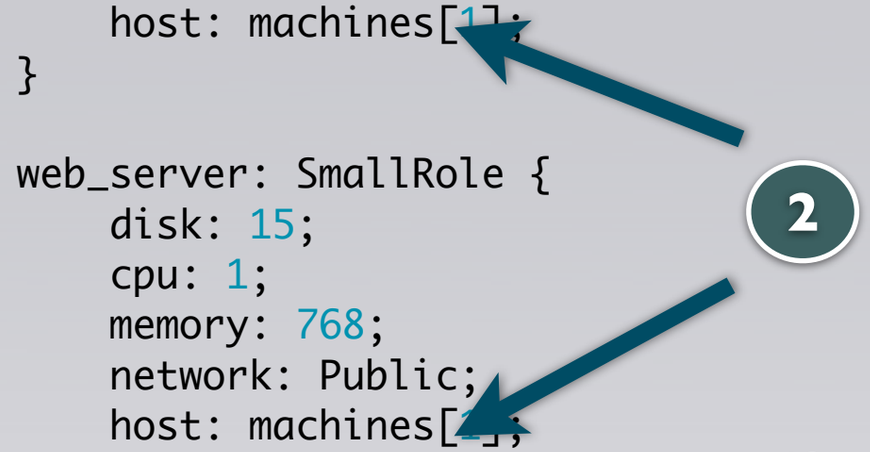
3

```
class Role {
  var host as ref Machine
```

Example (solution)




```
roles: Role {sql_server, web_server};

machines[1]: Machine {
    cpu: 16;
    memory: 16384;
    disk: 2048;
    network: Public;
}

machines[2]: Machine {
    cpu: 16;
    memory: 16384;
    disk: 2048;
    network: Public;
}

sql_server: LargeRole {
    disk: 412;
    cpu: 4;
    memory: 3584;
    network: Public;
    host: machines[1];
}

web_server: SmallRole {
    disk: 15;
    cpu: 1;
    memory: 768;
    network: Public;
    host: machines[2];
}
```



Optimisation

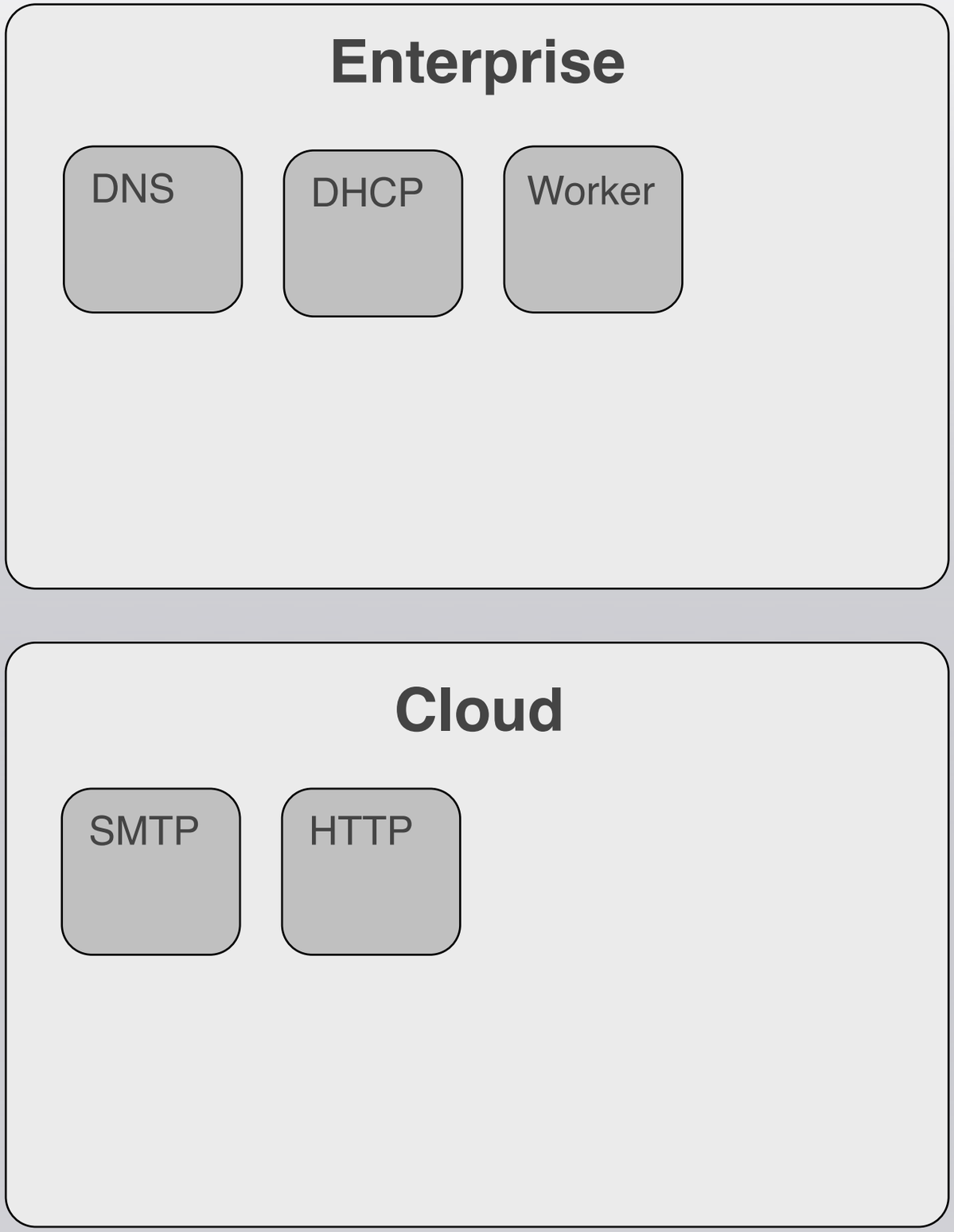
- Often want to optimise some aspect of the configuration
- or express (soft) preferences rather than (hard) constraints.
- MiniZinc and Gecode support maximisation of an objective function.

Example: Cloudbursting

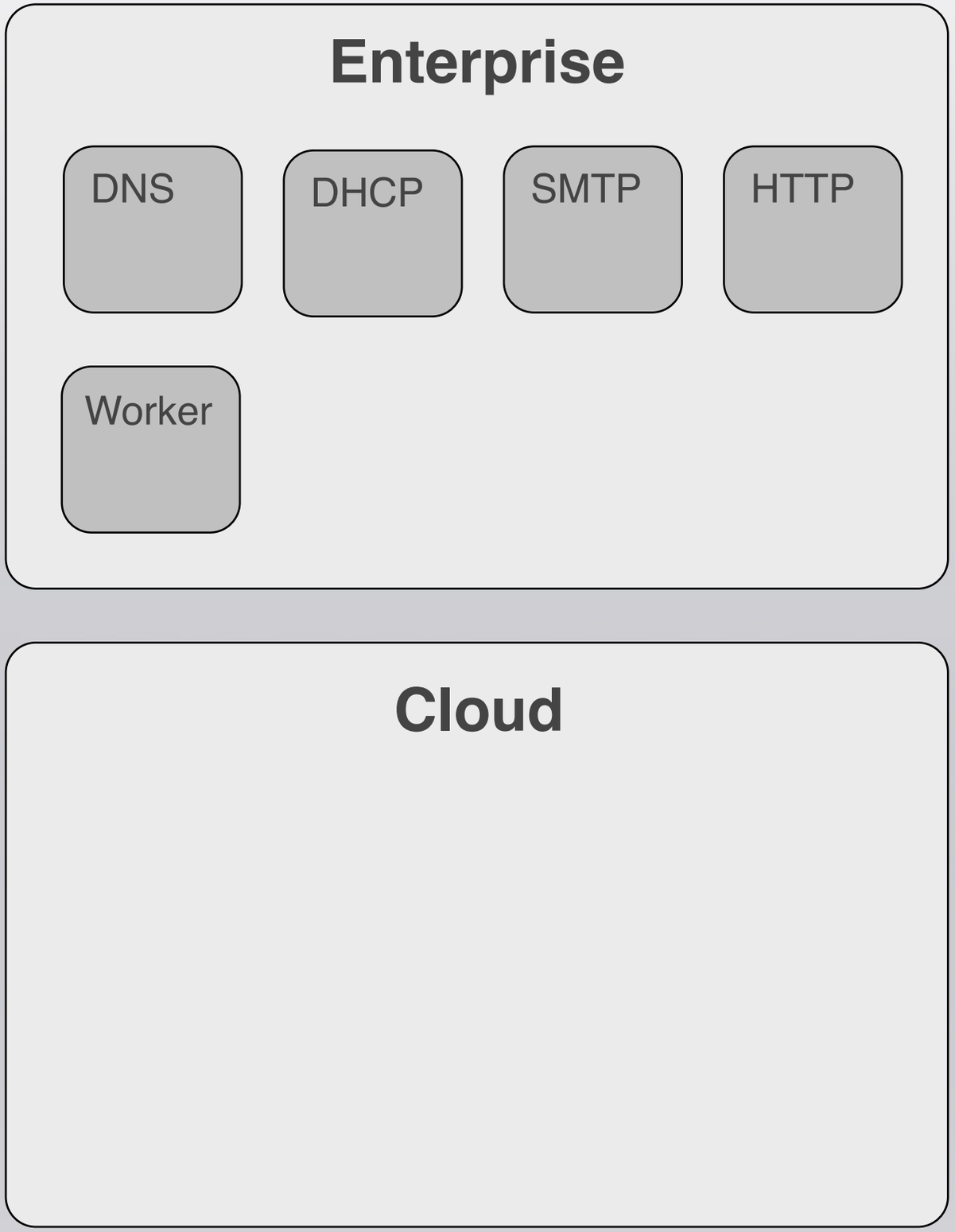
```
class Machine;

class Service {
    var host as ref Machine;
}

class Datacenter {
    var machines as Machine[8];
}

var cloud as Datacenter;
var enterprise as Datacenter;

var dhcp as Service[1];
var dns as Service[1];
var workers as Service[1];

// favour placement of machines in the enterprise datacenter
var utilization as int;
where utilization == count (s in services where s.host in enterprise.machines);

maximize utilization;
```

Without Optimisation

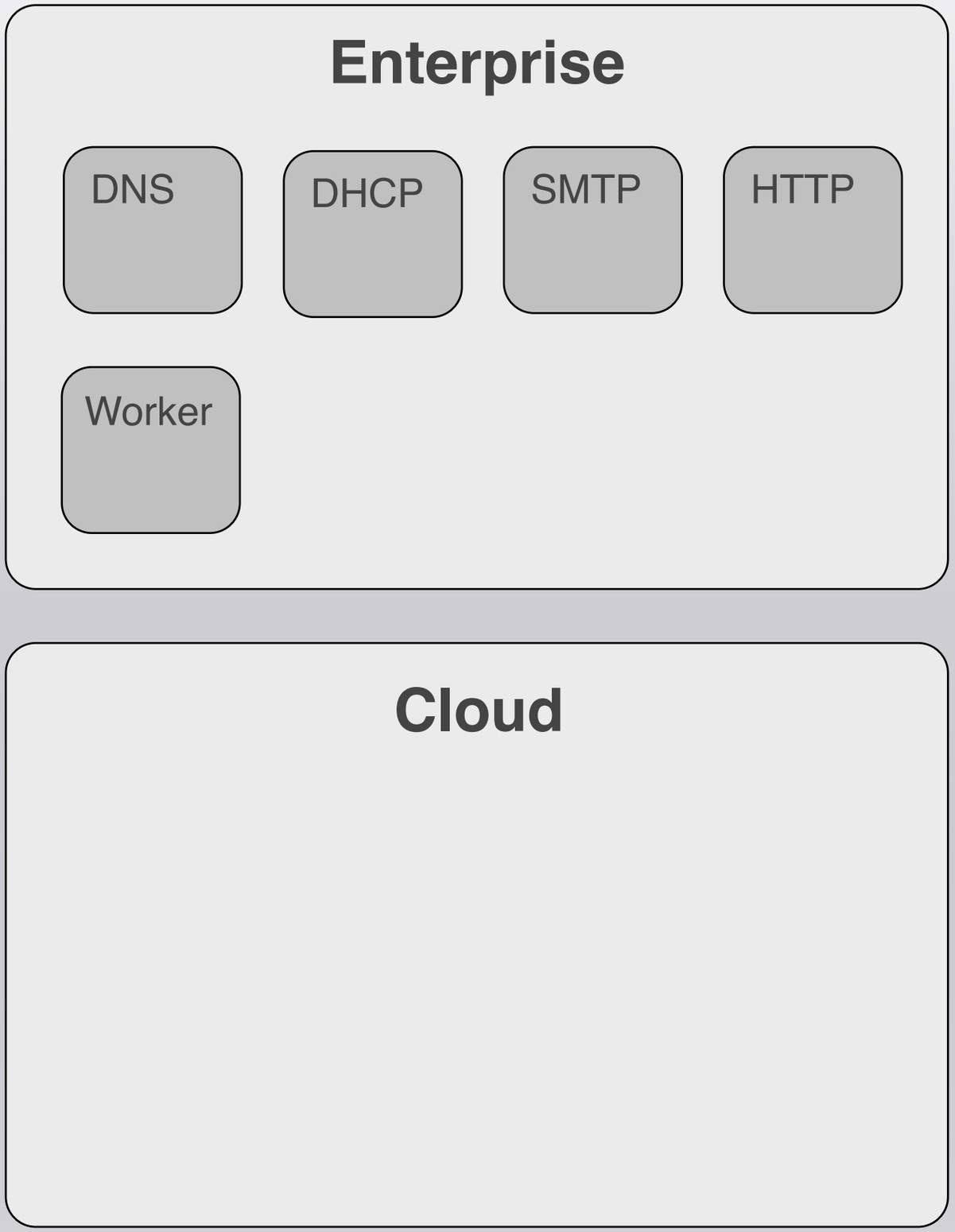
THE UNIVERSITY of EDINBURGH
informatics



With Optimisation



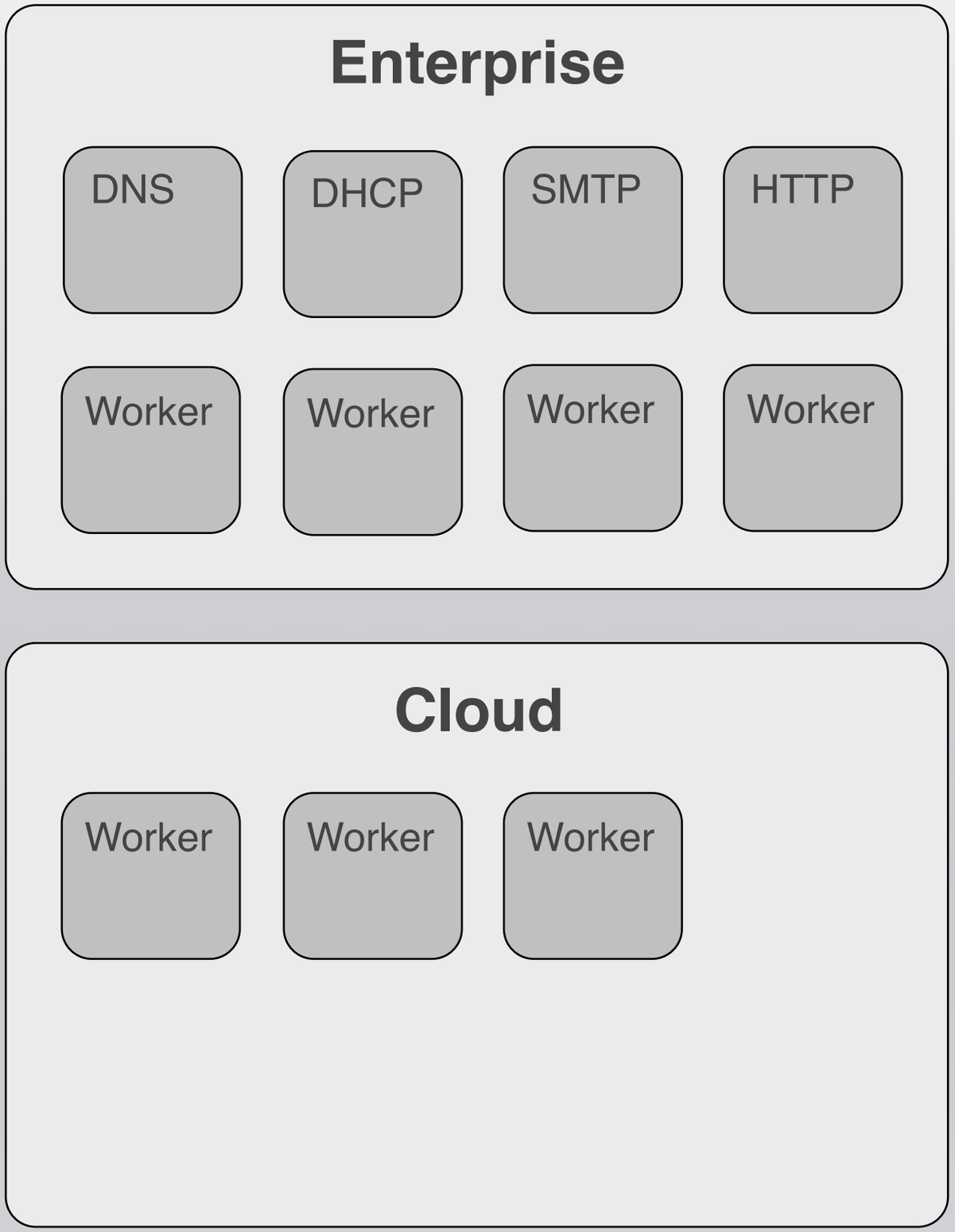
With Optimisation



With Optimisation

With Optimisation

With Optimisation

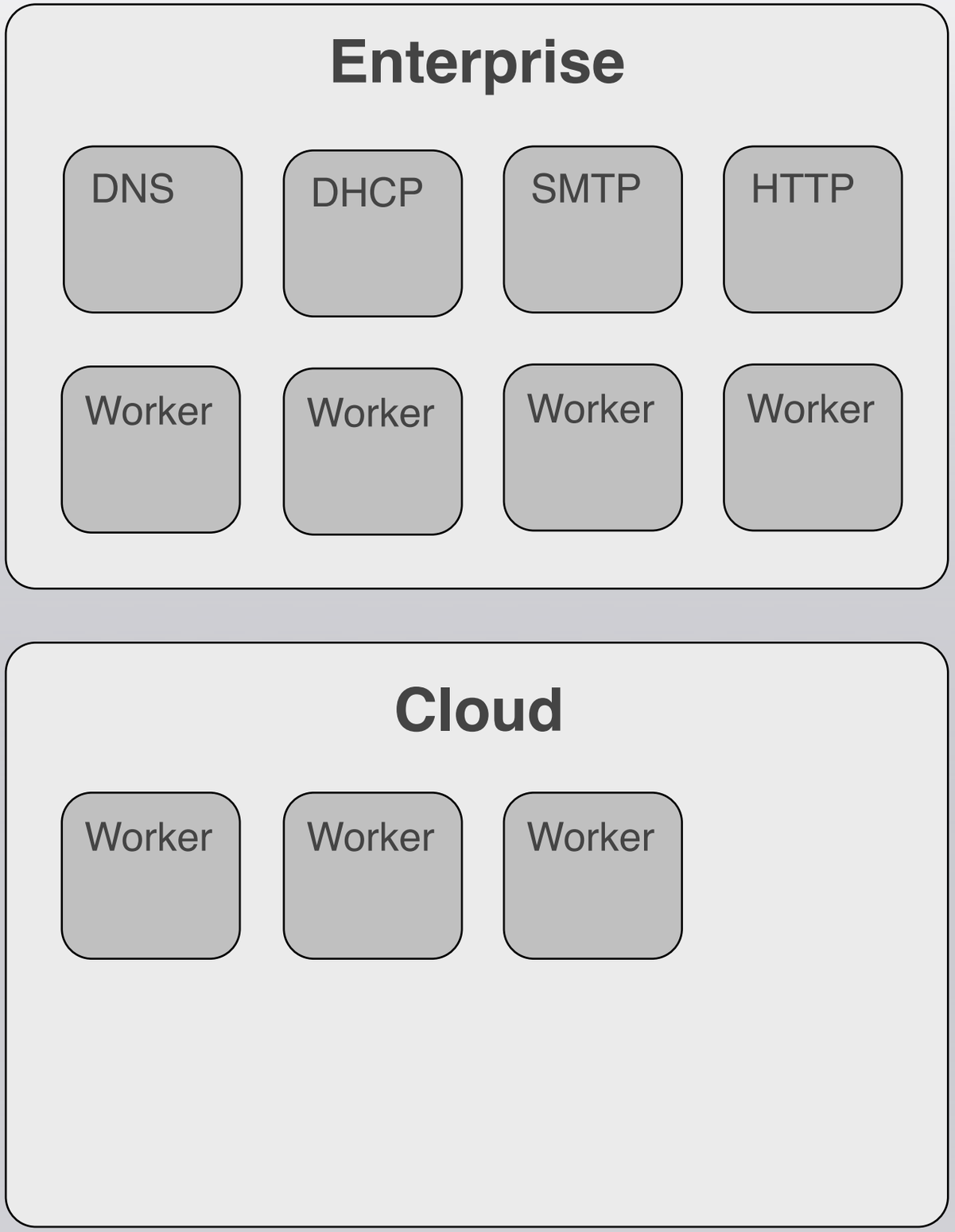


Re-Configuration

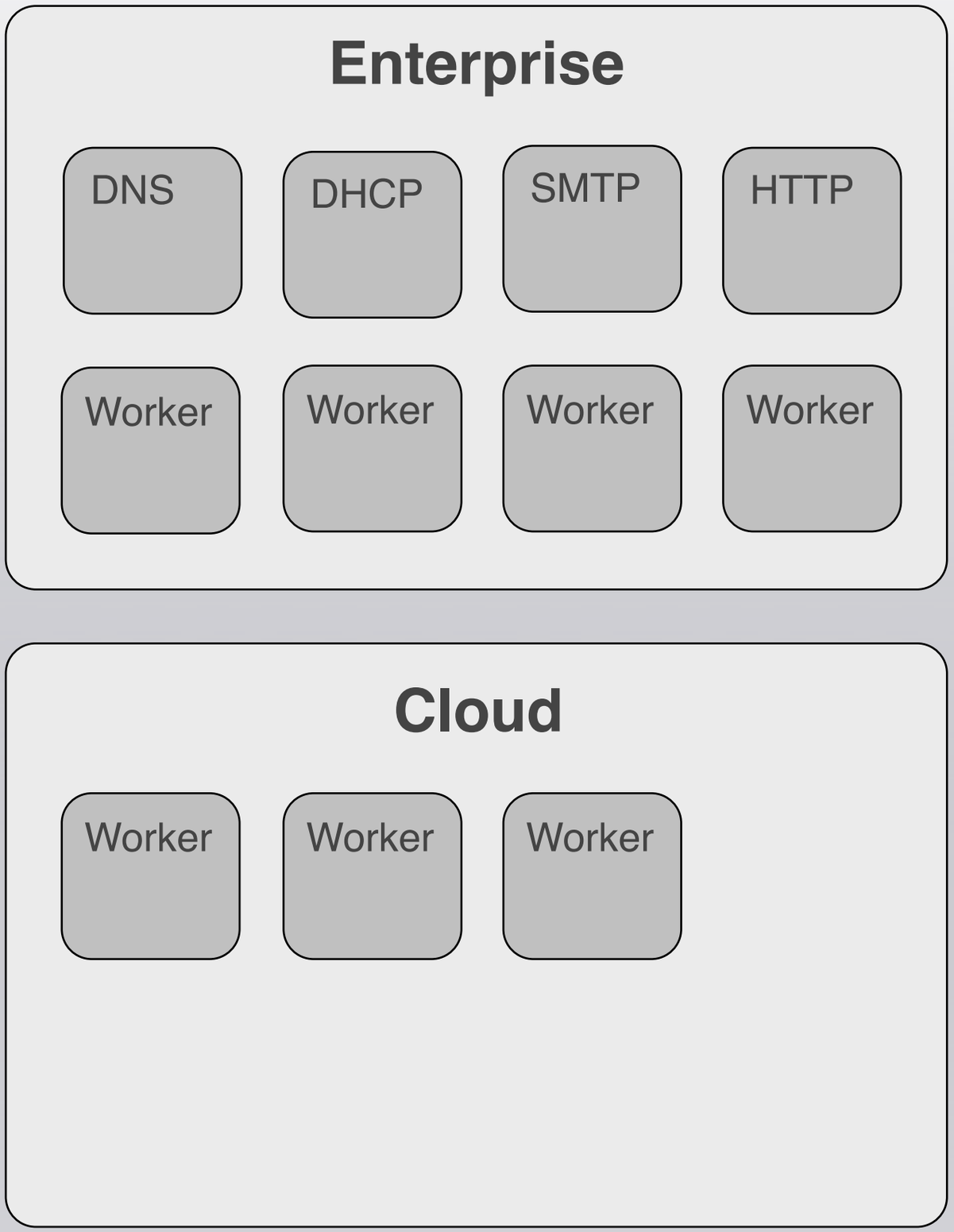
Re-Configuration

Re-Configuration

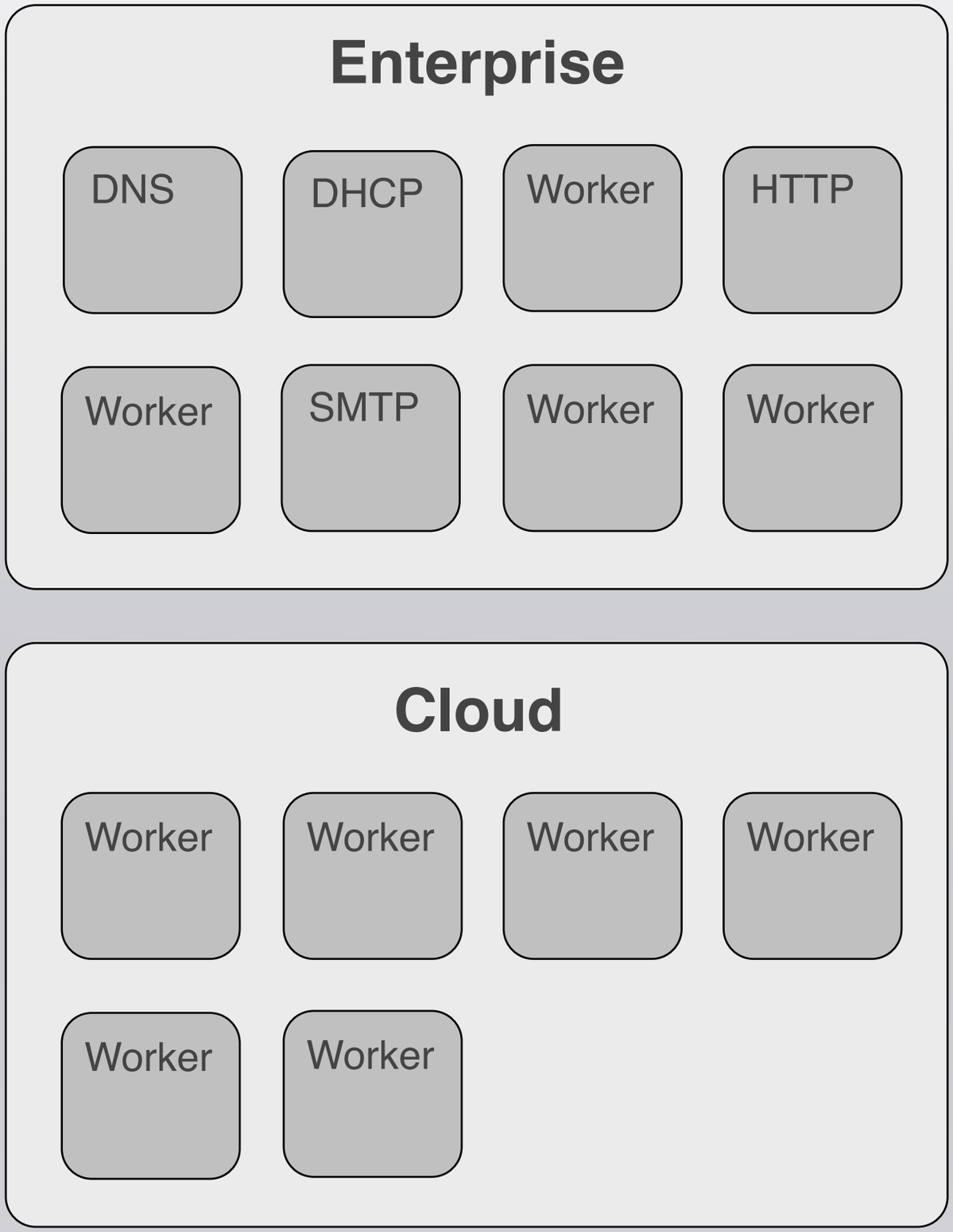
THE UNIVERSITY of EDINBURGH
informatics



Re-Configuration



Re-Configuration



Re-Configuration: Ongoing & Future Work

- Implementation based on system state, expressed as a an objective function in MiniZinc and solved with Gecode.
- Preference constraints without weighting
- ROADEF Challenge 2012 (Google Process Placement)
- Tradeoff between minimising changes and maximising objective function - e.g. scaling down of cloudbursting, when does the cost of maintaining the existing location of a server outweigh the cost of moving it?

THE UNIVERSITY *of* EDINBURGH
informatics

Microsoft®
Research

This work was funded by Microsoft Research through their
European PhD Scholarship Programme.