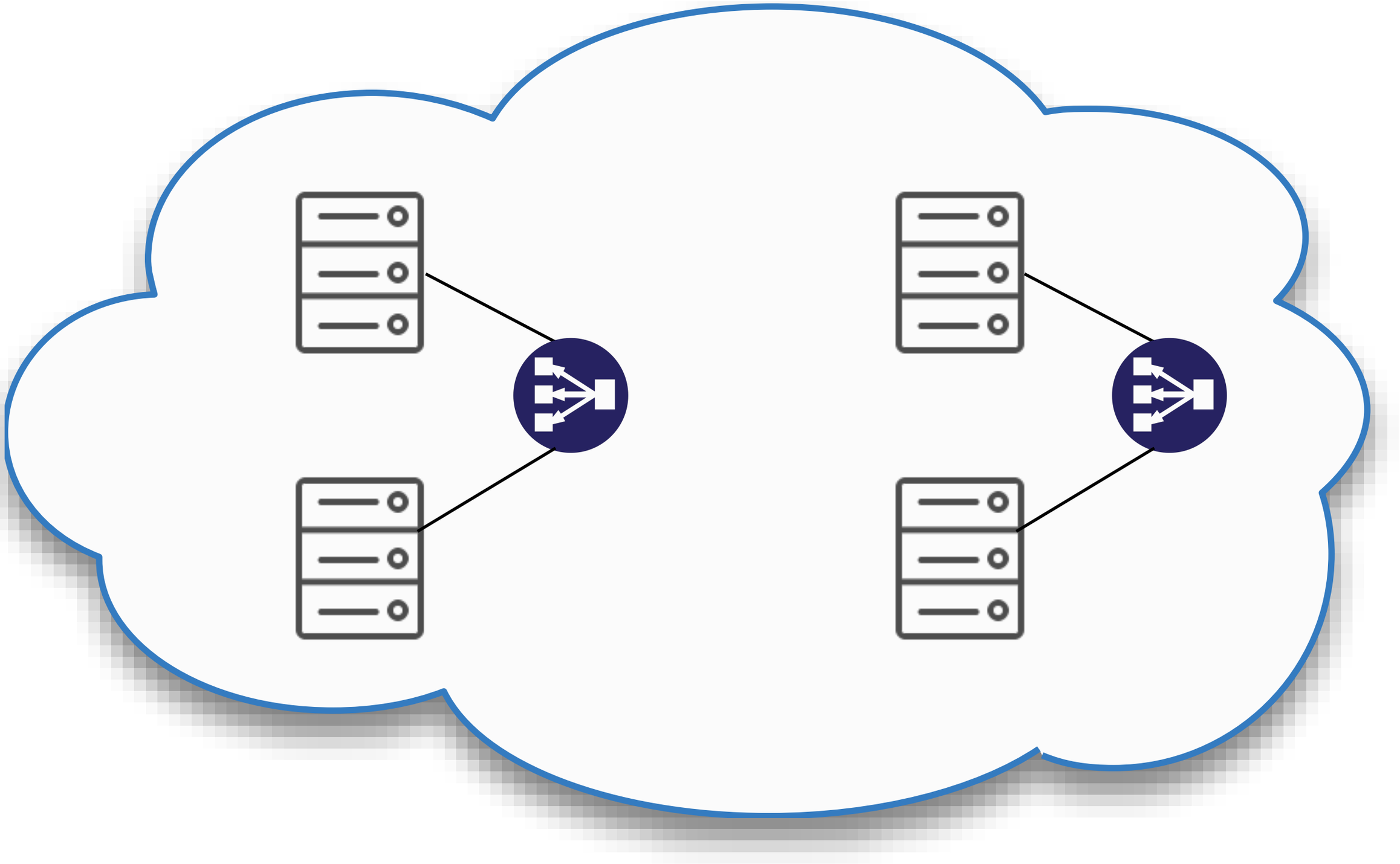
CPL: A Core Language for Cloud Computing

Oliver Bračevac
Sebastian Erdweg
Guido Salvaneschi
Mira Mezini

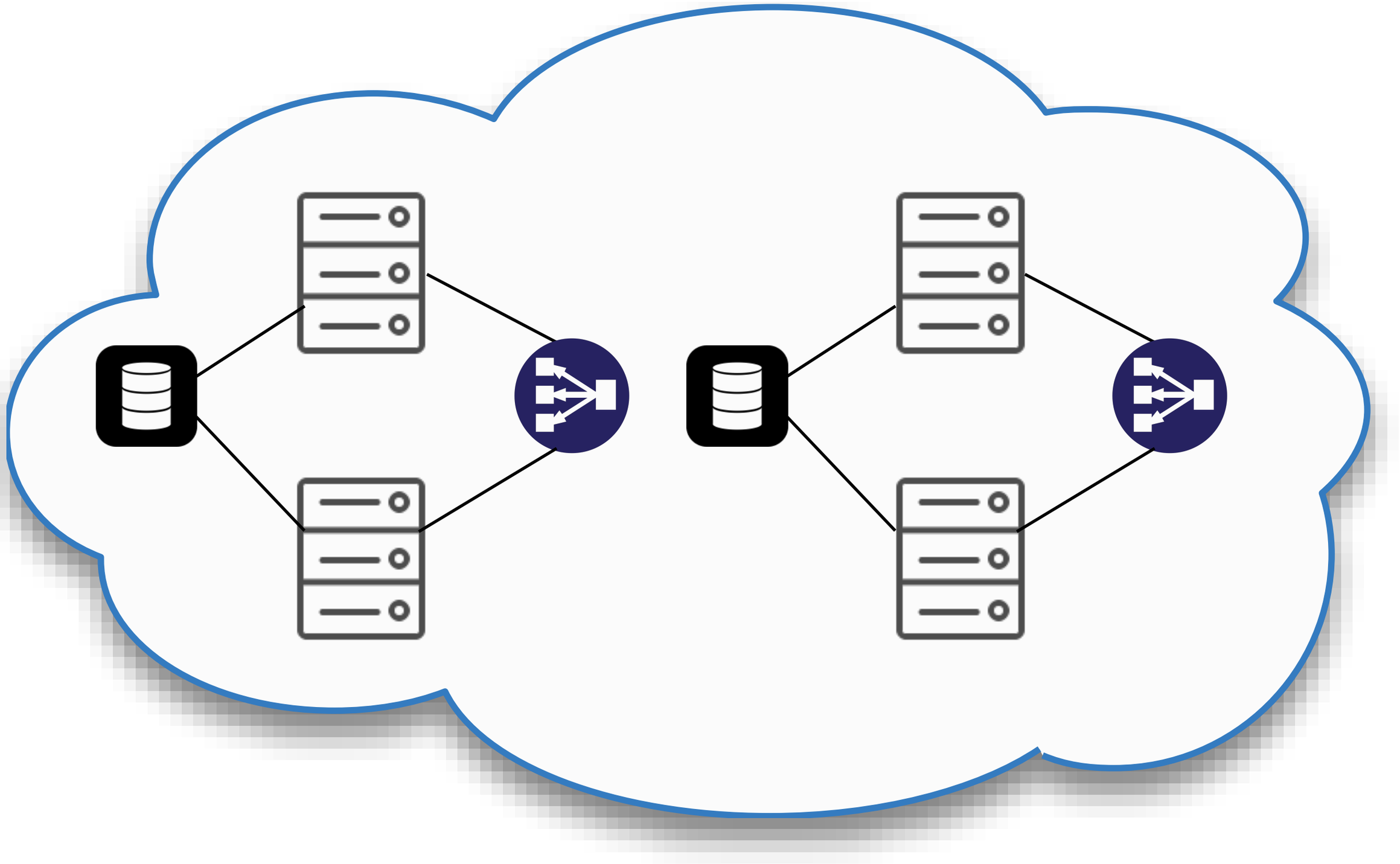
TECHNISCHE UNIVERSITÄT DARMSTADT
SOFTWARE TECHNOLOGY GROUP

Deployment in the Cloud

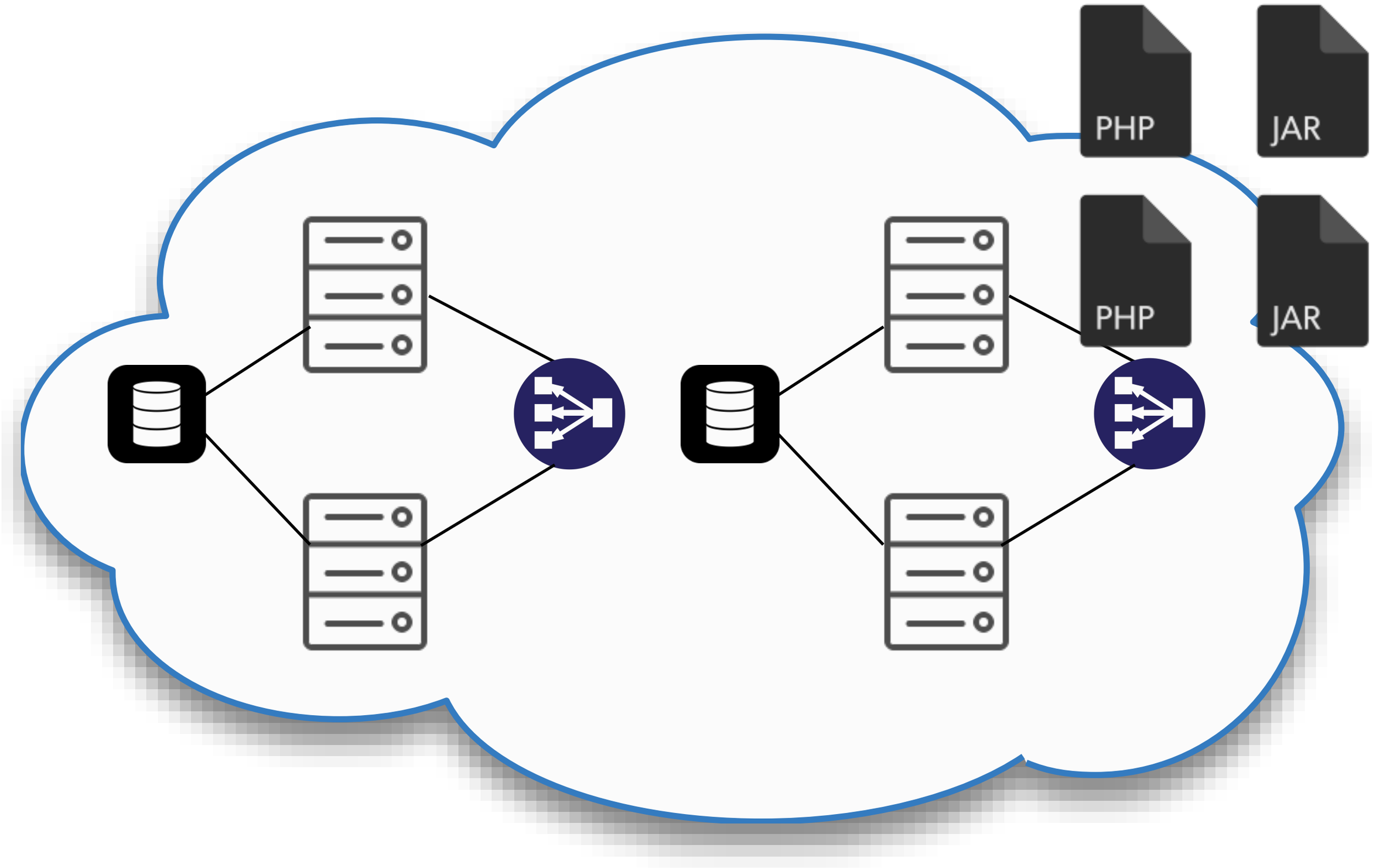
Deployment in the Cloud



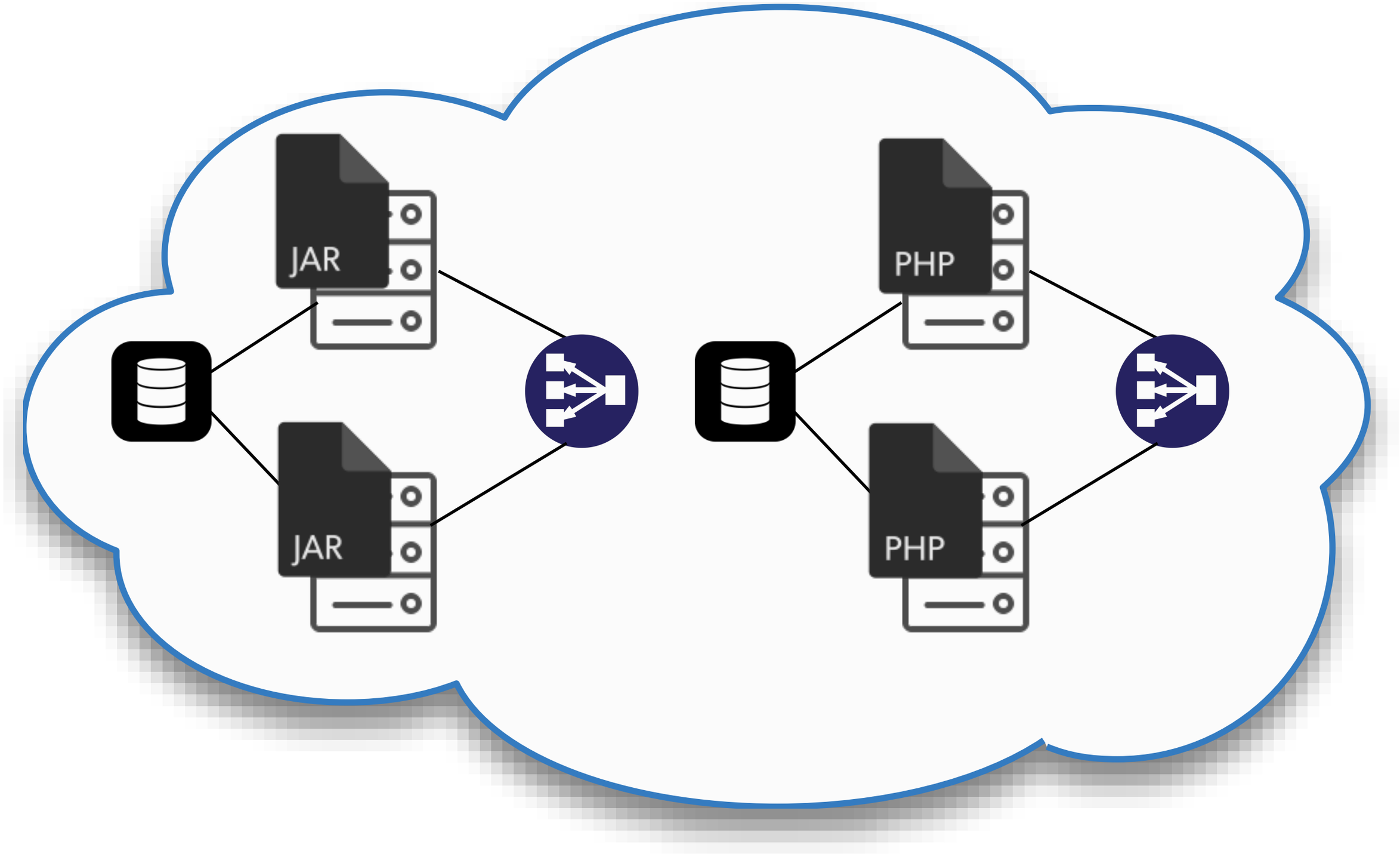
Deployment in the Cloud



Deployment in the Cloud



Deployment in the Cloud



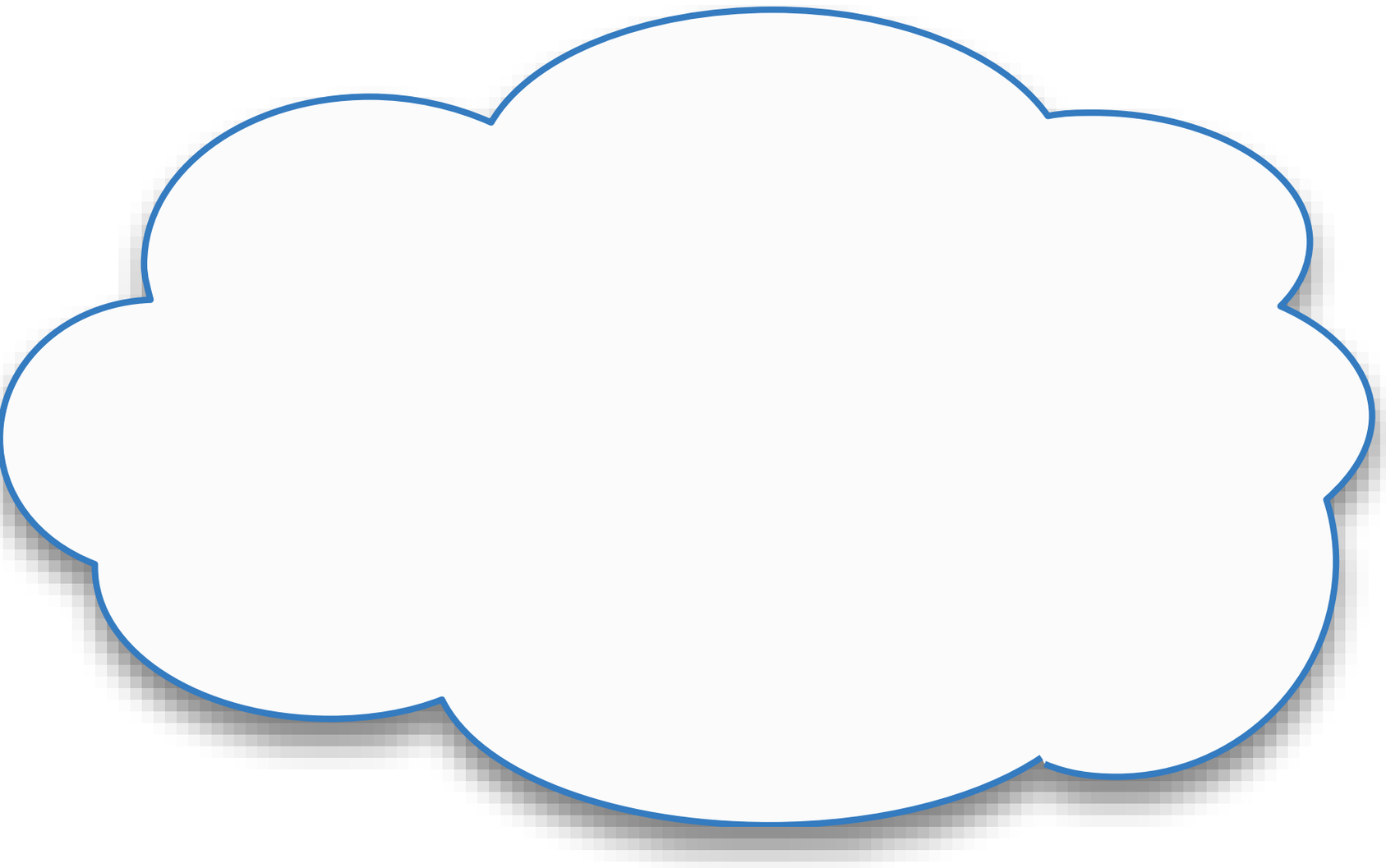
Deployment in the Cloud

Specify Deployment Patterns with modular DSLs


```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    }  //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "      --stack ", { "Ref" : "AWS::StackName" },
                    "      --resource WebServer ",
                    "      --configsets wordpress_install ",
                    "      --region ", { "Ref" : "AWS::Region" },"\n"
        ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Deployment in the Cloud

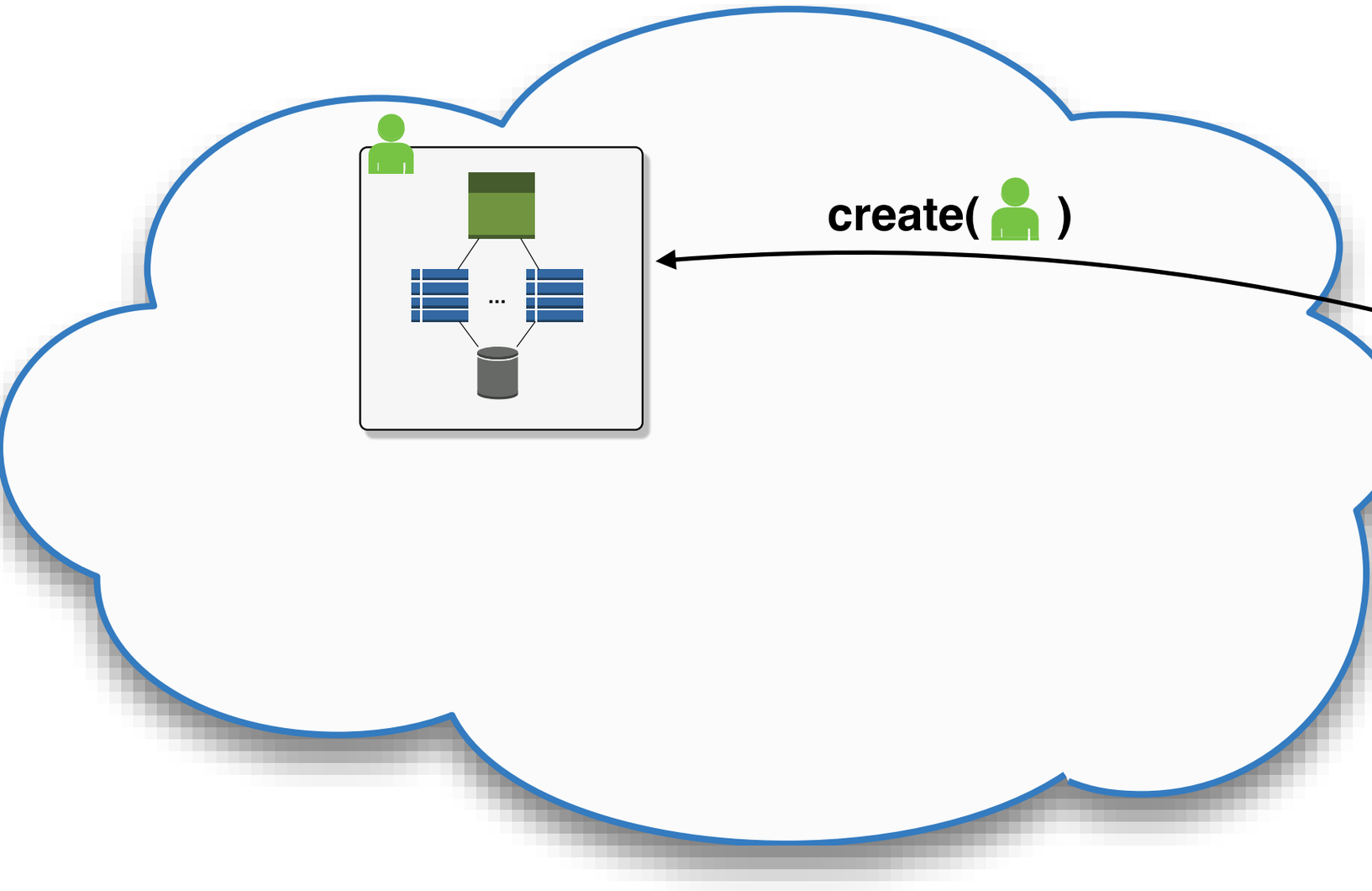
Specify Deployment Patterns with modular DSLs



```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    } //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "     --stack ", { "Ref" : "AWS::StackName" },
                    "     --resource WebServer ",
                    "     --configsets wordpress_install ",
                    "     --region ", { "Ref" : "AWS::Region" },"\n"
        ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Deployment in the Cloud

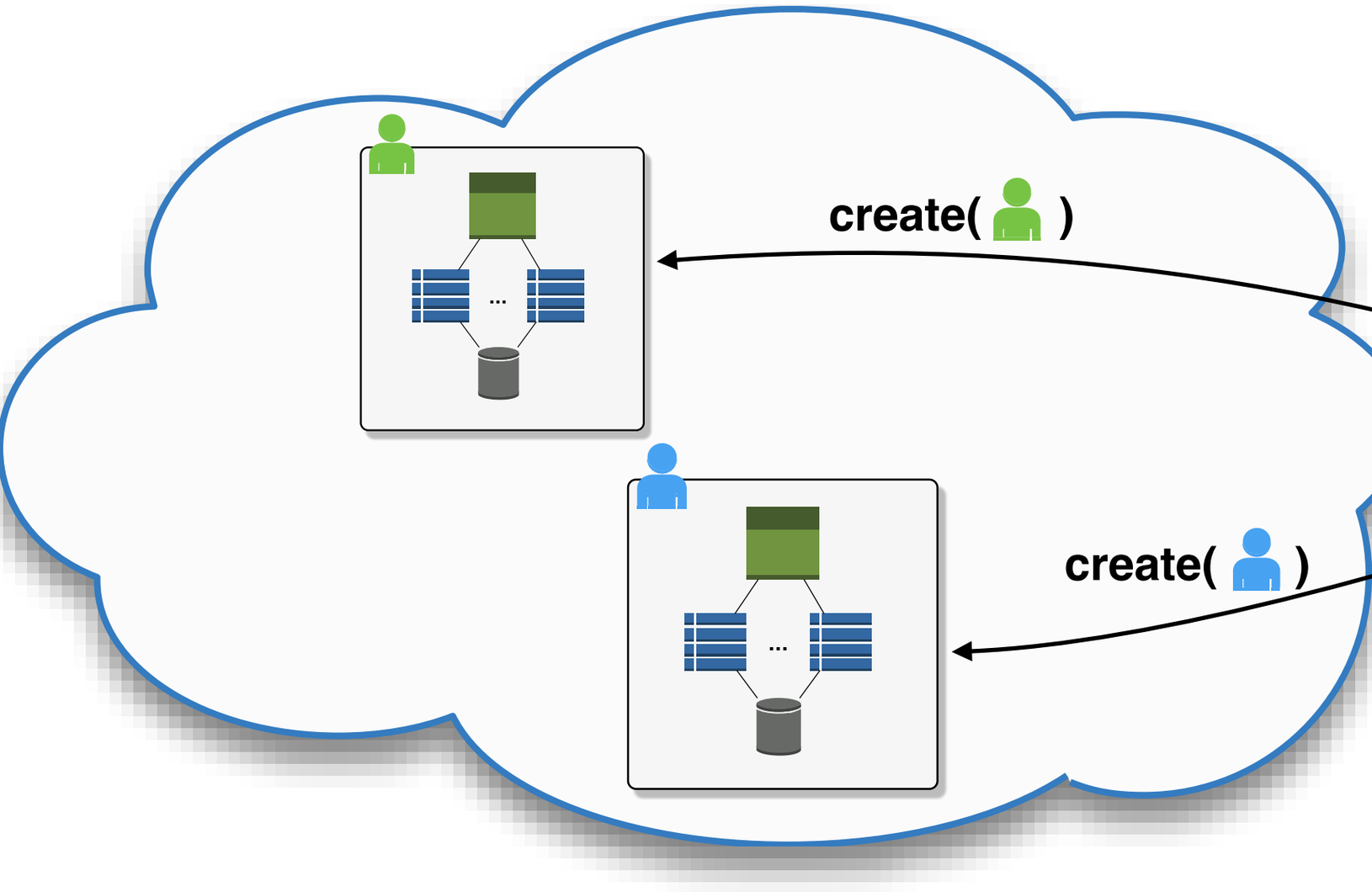
Specify Deployment Patterns with modular DSLs




```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    } //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "     --stack ", { "Ref" : "AWS::StackName" },
                    "     --resource WebServer ",
                    "     --configsets wordpress_install ",
                    "     --region ", { "Ref" : "AWS::Region" },"\n"
        ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Deployment in the Cloud

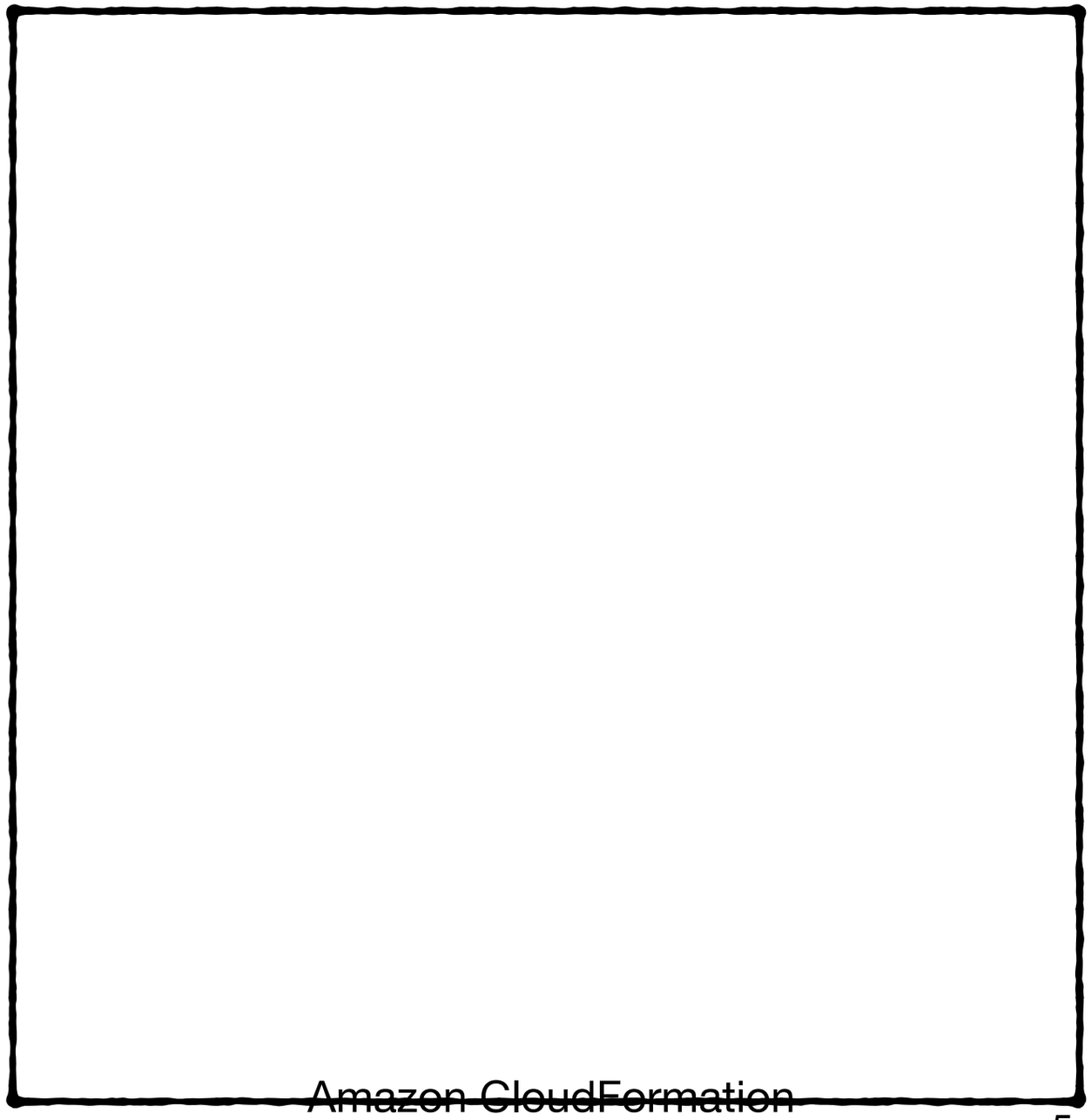
Specify Deployment Patterns with modular DSLs




```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    } //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "     --stack ", { "Ref" : "AWS::StackName" },
                    "     --resource WebServer ",
                    "     --configsets wordpress_install ",
                    "     --region ", { "Ref" : "AWS::Region" },"\n"
        ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Deployment DSLs

Anatomy of Templates



Deployment DSLs

Anatomy of Templates

- Parameters

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "t2.small",
    "AllowedValues" : ["t2.micro", "t2.small"]
  } //...
},
```

Deployment DSLs

Anatomy of Templates

- Parameters
- Resource Specifications

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "t2.small",
    "AllowedValues" : ["t2.micro", "t2.small"]
  } //...
},
"Resources" : {
  "WebServer": {
    "Type" : "AWS::EC2::Instance",
    "Properties": { ... },
  }
  "ElasticLoadBalancer" : {
   "Type":"AWS::ElasticLoadBalancing::LoadBalancer",
   "Properties" : { ... }
  }
},
```

Amazon CloudFormation

Deployment DSLs

Anatomy of Templates

- Parameters
- Resource Specifications
- Output

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "t2.small",
    "AllowedValues" : ["t2.micro", "t2.small"]
  } //...
},
"Resources" : {
  "WebServer": {
    "Type" : "AWS::EC2::Instance",
    "Properties": { ... },
  }
  "ElasticLoadBalancer" : {
   "Type":"AWS::ElasticLoadBalancing::LoadBalancer",
   "Properties" : { ... }
  }
},
"Outputs" : {
  "WebsiteURL" : {
    "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
    [ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]}}}
```

Deployment DSLs

Unsafe Parameterization

```
"files" : {
  "/var/www/php/index.php" : {
     "content" : {"Fn::Join" : [ "", [
       "<html>\n",
          "<head>\n",
            "</head>\n",
          "<body>\n",
            "<?php ", {"Ref" : "SetupFunction"}, "(); ?>\n",
          "</body>\n",
       "</html>\n" ]]},
  "mode" : "000500",
  "owner" : "root",
  "group" : "root"
  }
}
```

Deployment DSLs

Two-phase Staging: Indirect topology changes at run time

Deploy → Run

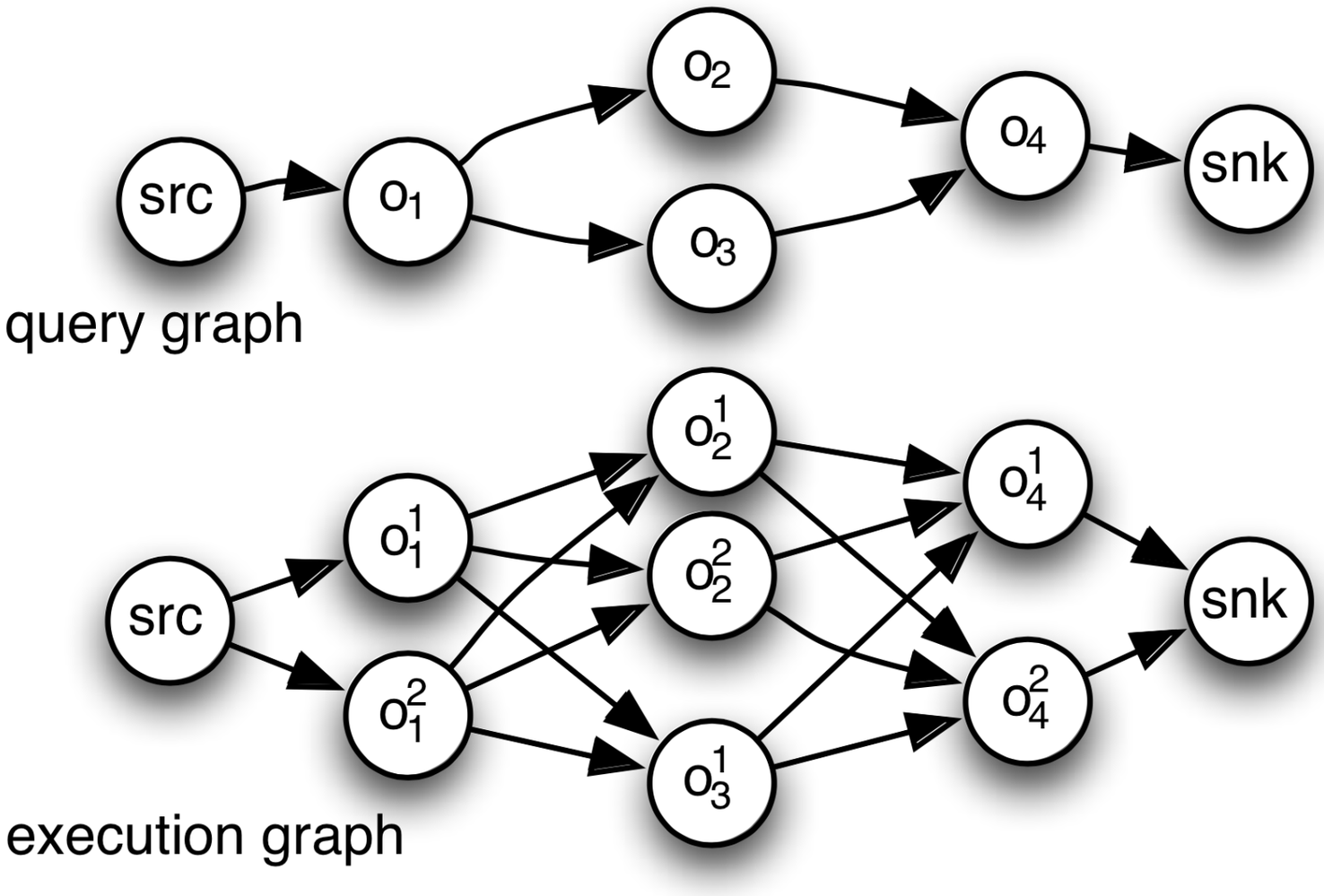


Image taken from [Fernandez et al. ’13]

Deployment DSLs

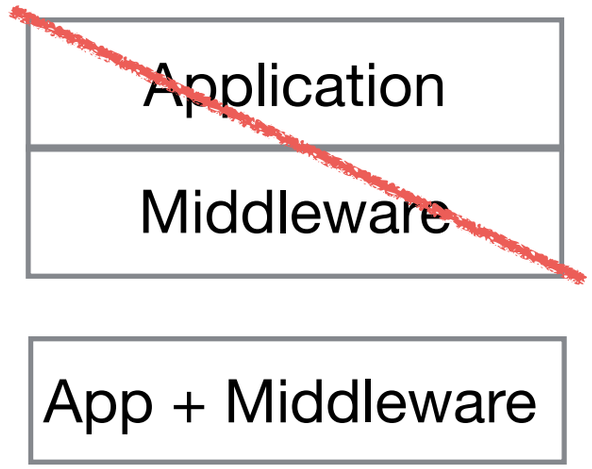
Inextensible Middleware Services

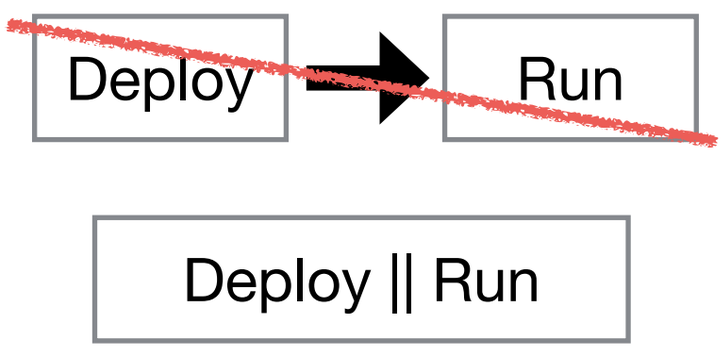
```
"ElasticLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "Instances" : [ {"Ref":"Ec2Instance1"},{"Ref":"Ec2Instance2"} ],
    "Listeners" : [ {
      "LoadBalancerPort" : "80",
      "InstancePort" : {"Ref":"WebServerPort"},
      "Protocol" : "HTTP"
    } ],
   ...
 }
}
```

The Cloud Platform Language

A core language for reasoning & studying

- Distributed cloud applications
- Programmable, extensible middleware
- Safe service composition
- Dynamic deployment





First-class Servers

$$
\begin{array}{l}
\texttt{Fib} = \mathbf{srv}\ \{ \\
\quad \texttt{fib}\langle n,\ k\rangle \ \triangleright\ \mathbf{this}\sharp\texttt{acc}\langle n, \texttt{0}, \texttt{1}, k\rangle \\
\\
\quad \texttt{acc}\langle n,\ a,\ b,\ k\rangle\ \triangleright \\
\qquad \mathbf{if}\ n = \texttt{0} \\
\qquad \mathbf{then}\ k\langle a\rangle \\
\qquad \mathbf{else}\ \ \mathbf{this}\sharp\texttt{acc}\langle n - \texttt{1},\ b,\ a + b,\ k\rangle \\
\}
\end{array}
$$

$$
(\mathbf{spwn}\ \texttt{Fib})\sharp\texttt{fib}\langle \texttt{10},k\rangle \longrightarrow i\sharp\texttt{fib}\langle \texttt{10},k\rangle \longrightarrow i\sharp\texttt{acc}\langle \texttt{10},\texttt{0},\texttt{1},k\rangle \longrightarrow^{*} k\langle \texttt{55}\rangle
$$

First-class Servers

$$
\begin{array}{l}
\texttt{Fib} = \mathbf{srv}\ \{ \\
\quad \texttt{fib}\langle n,\ k\rangle \triangleright \mathbf{this}\sharp\texttt{acc}\langle n, 0, 1, k\rangle \\
\\
\quad \texttt{acc}\langle n,\ a,\ b,\ k\rangle \triangleright \\
\qquad \mathbf{if}\ n = 0 \\
\qquad \mathbf{then}\ k\langle a\rangle \\
\qquad \mathbf{else}\ \ \mathbf{this}\sharp\texttt{acc}\langle n-1,\ b,\ a+b,\ k\rangle \\
\}
\end{array}
$$

Server Template

$$(\mathbf{spwn}\ \texttt{Fib})\sharp\texttt{fib}\langle 10,k\rangle \longrightarrow i\sharp\texttt{fib}\langle 10,k\rangle \longrightarrow i\sharp\texttt{acc}\langle 10,0,1,k\rangle \longrightarrow^* k\langle 55\rangle$$

First-class Servers

$$
\begin{array}{l}
\texttt{Fib} = \textbf{srv}\ \{ \\
\quad \texttt{fib}\langle n,\ k\rangle \triangleright \textbf{this}\sharp\texttt{acc}\langle n, 0, 1, k\rangle \\
\\
\quad \texttt{acc}\langle n,\ a,\ b,\ k\rangle \triangleright \\
\qquad \textbf{if}\ n = 0 \\
\qquad \textbf{then}\ k\langle a\rangle \\
\qquad \textbf{else}\ \ \textbf{this}\sharp\texttt{acc}\langle n-1,\ b,\ a+b,\ k\rangle \\
\}
\end{array}
$$

Server Template

$$(\textbf{spwn}\ \texttt{Fib})\sharp\texttt{fib}\langle 10,k\rangle \longrightarrow i\sharp\texttt{fib}\langle 10,k\rangle \longrightarrow i\sharp\texttt{acc}\langle 10,0,1,k\rangle \longrightarrow^{*} k\langle 55\rangle$$

Cloud spawns server instances from templates

Message Passing & CPS

```
Fib2 = srv {
   fib⟨n, k⟩ ▷
      if n = 0 ∨ n = 1
         k⟨n⟩
      else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
              in k⟨x + y⟩
}
```

Message Passing & CPS

```
Fib2 = srv {
   fib⟨n, k⟩ ▷
      if n = 0 ∨ n = 1
      k⟨n⟩
   else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```

Direct style program

Message Passing & CPS

```
Fib2 = srv {
   fib⟨n, k⟩ ▷
      if n = 0 ∨ n = 1
      k⟨n⟩
   else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```

Direct style program

Desugaring:

$$\textbf{letk}\ x = e_1\langle \overline{e} \rangle\ \textbf{in}\ e_2 \quad \rightsquigarrow \quad e_1\langle \overline{e}, (\textbf{spwn}\ (\textbf{srv}\ \mathtt{k}\langle x \rangle \triangleright e_2))\sharp \mathtt{k} \rangle$$

Message Passing & CPS

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```

Direct style program

Desugaring:

Callback to server

$$\textbf{letk}\ x = e_1\langle \overline{e} \rangle\ \textbf{in}\ e_2 \quad \rightsquigarrow \quad e_1\langle \overline{e}, (\textbf{spwn}\ (\textbf{srv}\ \mathtt{k}\langle x \rangle \triangleright e_2))\sharp \mathtt{k} \rangle$$

There is only async message passing!

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩
  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

```
Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

```
Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}
```

```
let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

```
Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}
```

```
let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

```
Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}
```

```
let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}
```

```
Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

    // details omitted ...
}
```

```
let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}

Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

    // details omitted ...
}

let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}

Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

    // details omitted ...
}

let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}

Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}

let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

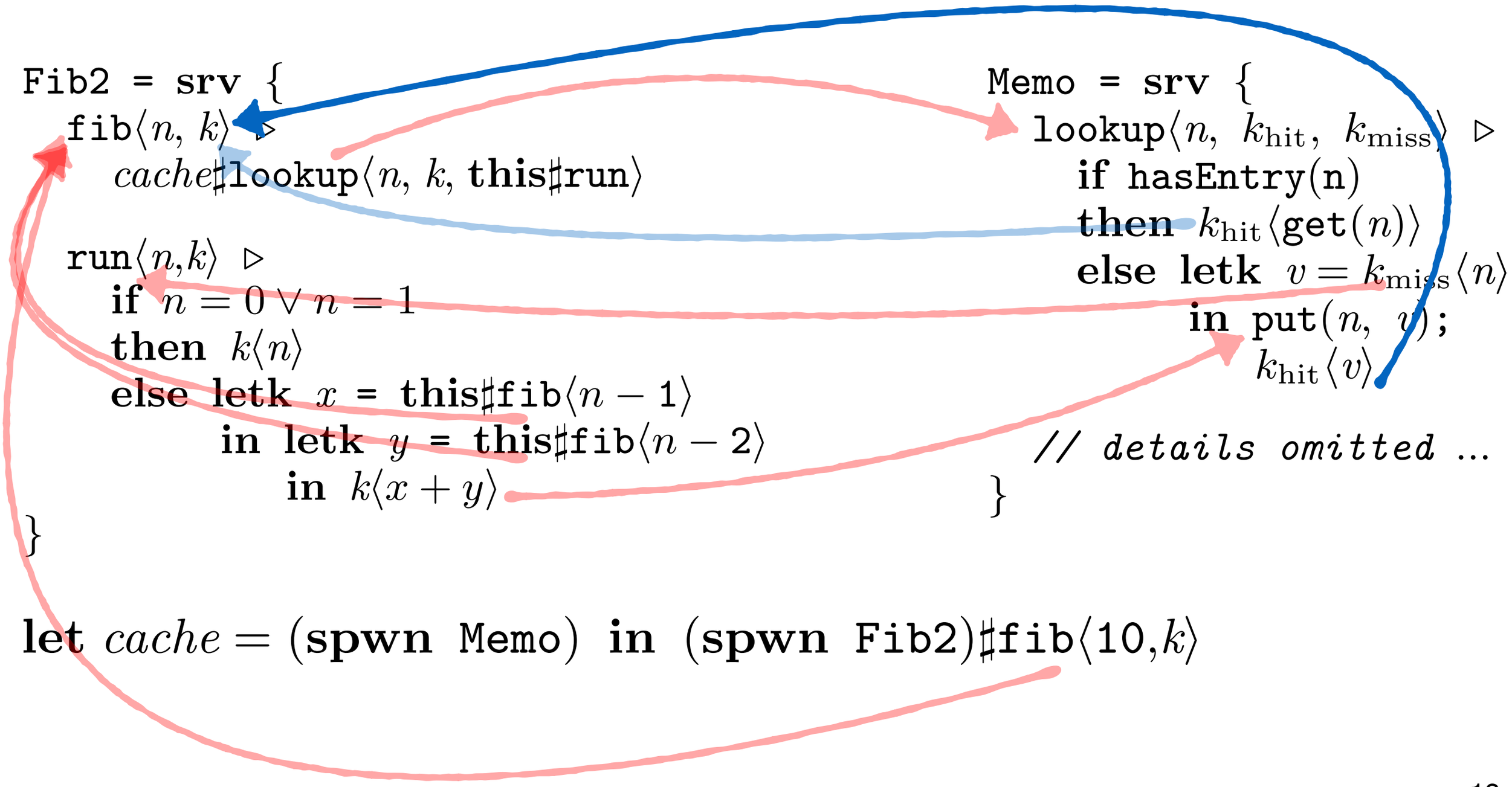
Message Passing & CPS

Service Composition

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache♯lookup⟨n, k, this♯run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
    then k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
           in letk y = this♯fib⟨n − 2⟩
                in k⟨x + y⟩
}

Memo = srv {
  lookup⟨n, k_hit, k_miss⟩ ▷
    if hasEntry(n)
    then k_hit⟨get(n)⟩
    else letk v = k_miss⟨n⟩
           in put(n, v);
              k_hit⟨v⟩

  // details omitted ...
}

let cache = (spwn Memo) in (spwn Fib2)♯fib⟨10,k⟩
```

Message Passing & CPS

Service Composition



Concurrency & Join Patterns

[Fournet and Gonthier ’96, Turon and Russo ‘11]

```
Fib3 = srv {
  fib⟨n, k⟩ ▷
    this♯caller⟨k⟩ ‖ this♯invoke⟨n⟩

  invoke⟨n⟩ ▷
    if n = 0 ∨ n = 1
       this♯res⟨n⟩
    else   (spwn Fib3)♯fib⟨n − 1, this♯left⟩
         ‖(spwn Fib3)♯fib⟨n − 2, this♯right⟩

  left⟨n⟩ & right⟨m⟩ ▷
    this♯res⟨n + m⟩

  res⟨n⟩ & caller⟨k⟩ ▷ k⟨n⟩
}
```

Concurrency & Join Patterns

[Fournet and Gonthier ’96, Turon and Russo ‘11]

```
Fib3 = srv {
  fib⟨n, k⟩ ▷
    this♯caller⟨k⟩ ‖ this♯invoke⟨n⟩

  invoke⟨n⟩ ▷
    if n = 0 ∨ n = 1
      this♯res⟨n⟩
    else (spwn Fib3)♯fib⟨n − 1, this♯left⟩
       ‖(spwn Fib3)♯fib⟨n − 2, this♯right⟩

  left⟨n⟩ & right⟨m⟩ ▷
    this♯res⟨n + m⟩

  res⟨n⟩ & caller⟨k⟩ ▷ k⟨n⟩
}
```

Parallel execution

Concurrency & Join Patterns

[Fournet and Gonthier '96, Turon and Russo '11]

```
Fib3 = srv {
  fib⟨n, k⟩ ▷
    this♯caller⟨k⟩ ‖ this♯invoke⟨n⟩

  invoke⟨n⟩ ▷
    if n = 0 ∨ n = 1
      this♯res⟨n⟩
    else (spwn Fib3)♯fib⟨n − 1, this♯left⟩
       ‖(spwn Fib3)♯fib⟨n − 2, this♯right⟩

  left⟨n⟩ & right⟨m⟩ ▷
    this♯res⟨n + m⟩

  res⟨n⟩ & caller⟨k⟩ ▷ k⟨n⟩
}
```

Parallel execution

Joins for synchronization

Server Management

```
Migratable = srv {
  migrate⟨dest⟩ ▷
    let img = snap this
    in (dest⟨img⟩
        ‖ repl this 0)

  // ...
}
```

Server Management

```
Migratable = srv {
  migrate⟨dest⟩ ▷
    let img = snap this
    in (dest⟨img⟩
        ‖ repl this 0)

  // ...
}
```

Retrieve server image from cloud

Server Management

```
Migratable = srv {
  migrate⟨dest⟩ ▷
    let img = snap this
    in (dest⟨img⟩
        ‖ repl this 0)
  // ...
}
```

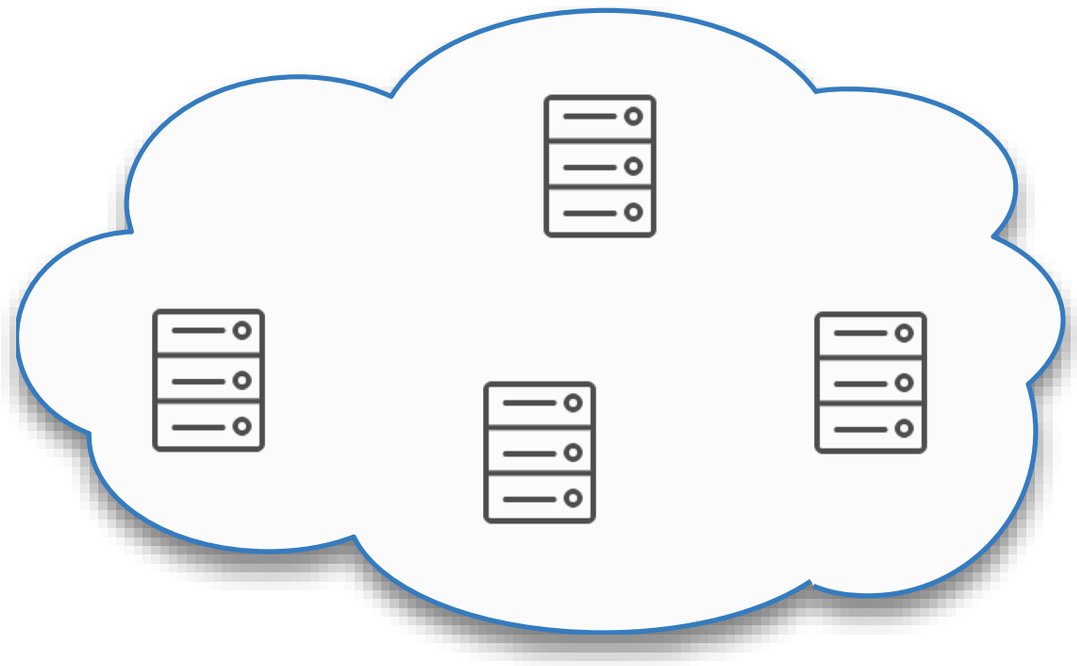
Retrieve server image from cloud

Replace with different image (inactive server in this case)

Server Placement

Are First-Class Servers VMs?

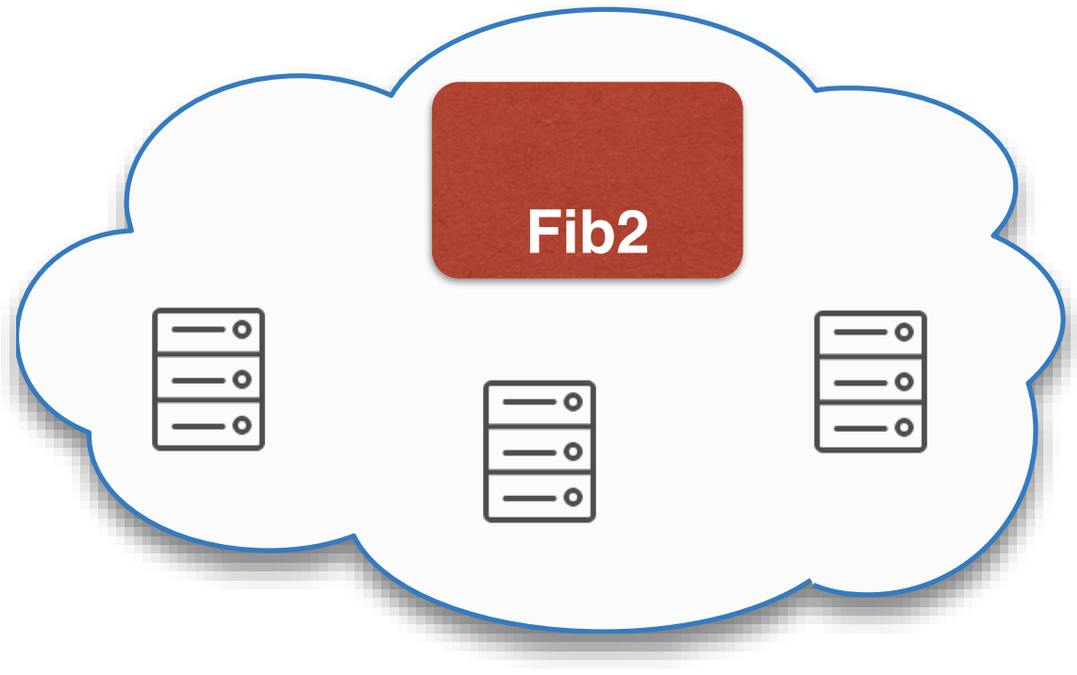
```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    if n = 0 ∨ n = 1
      k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```



Server Placement

Are First-Class Servers VMs?

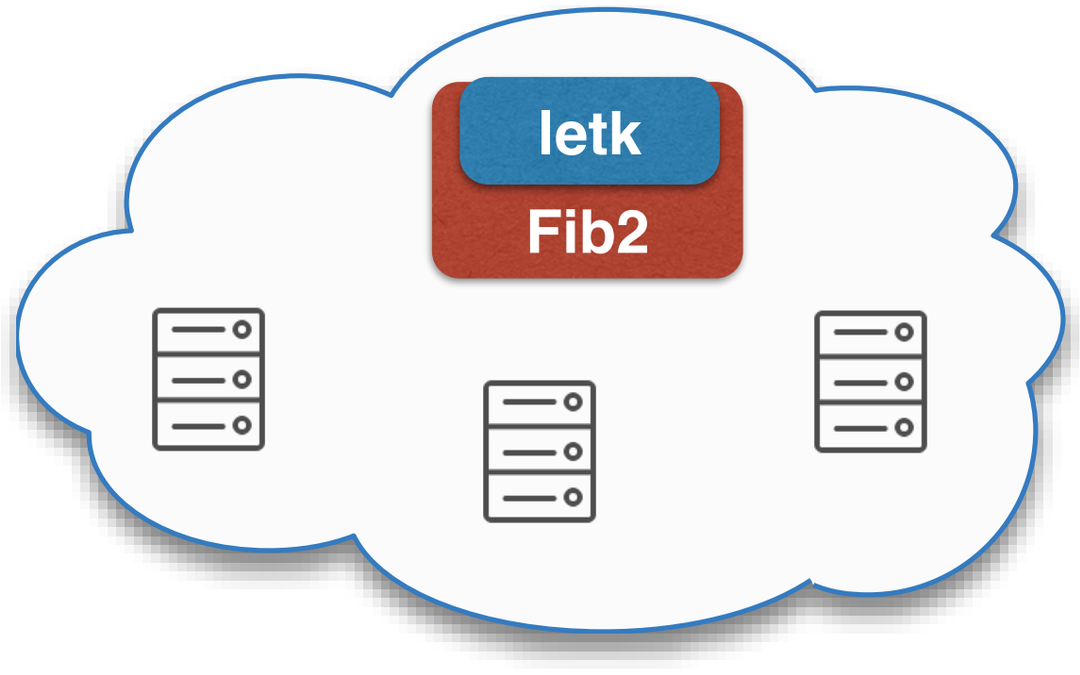
```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    if n = 0 ∨ n = 1
      k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```



Server Placement

Are First-Class Servers VMs?

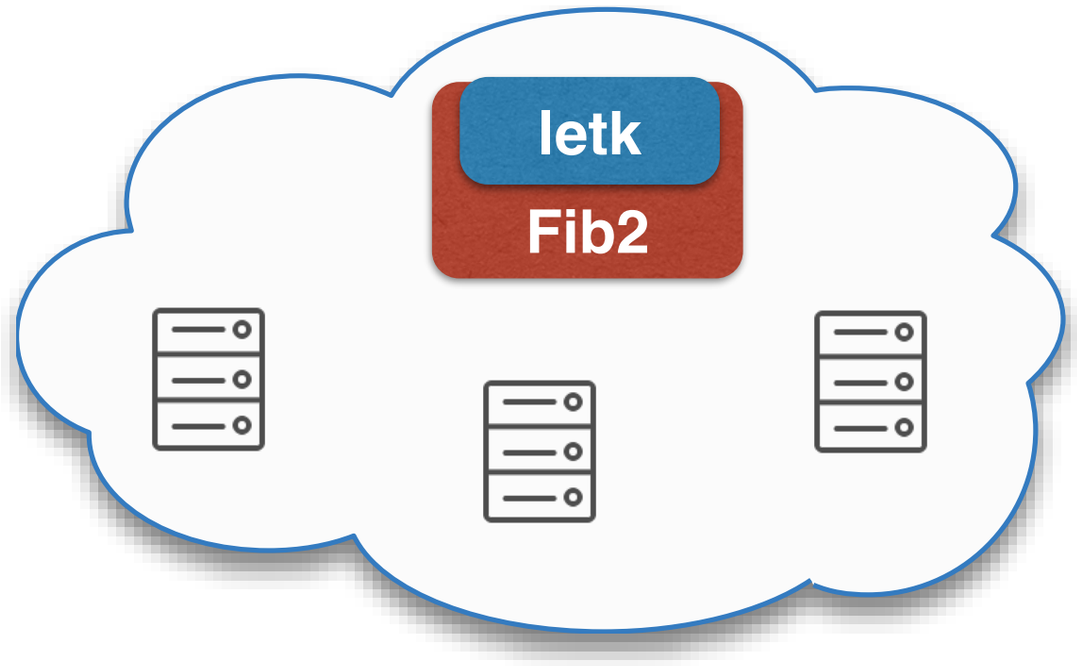
```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    if n = 0 ∨ n = 1
      k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```



Server Placement

Are First-Class Servers VMs?

```
Fib2 = srv {
  fib⟨n, k⟩ ▷
    if n = 0 ∨ n = 1
      k⟨n⟩
    else letk x = this♯fib⟨n − 1⟩
         in letk y = this♯fib⟨n − 2⟩
            in k⟨x + y⟩
}
```



Current implementation: Annotations for locality

Future work: Automated placement by run time

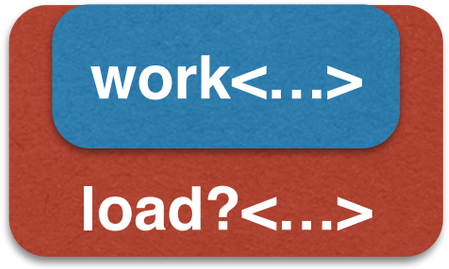
Server & Service Combinators

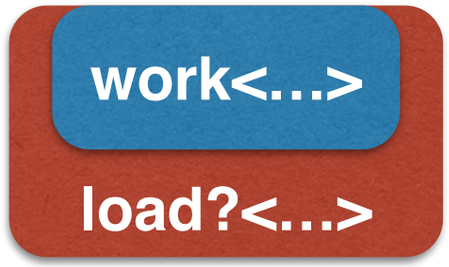
Server & Service Combinators

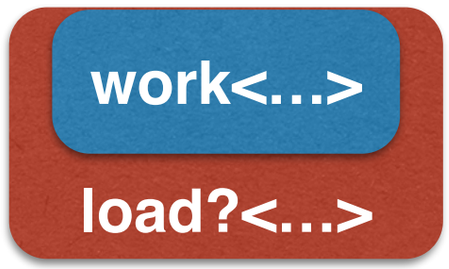
Load Balancing

Server & Service Combinators

Load Balancing

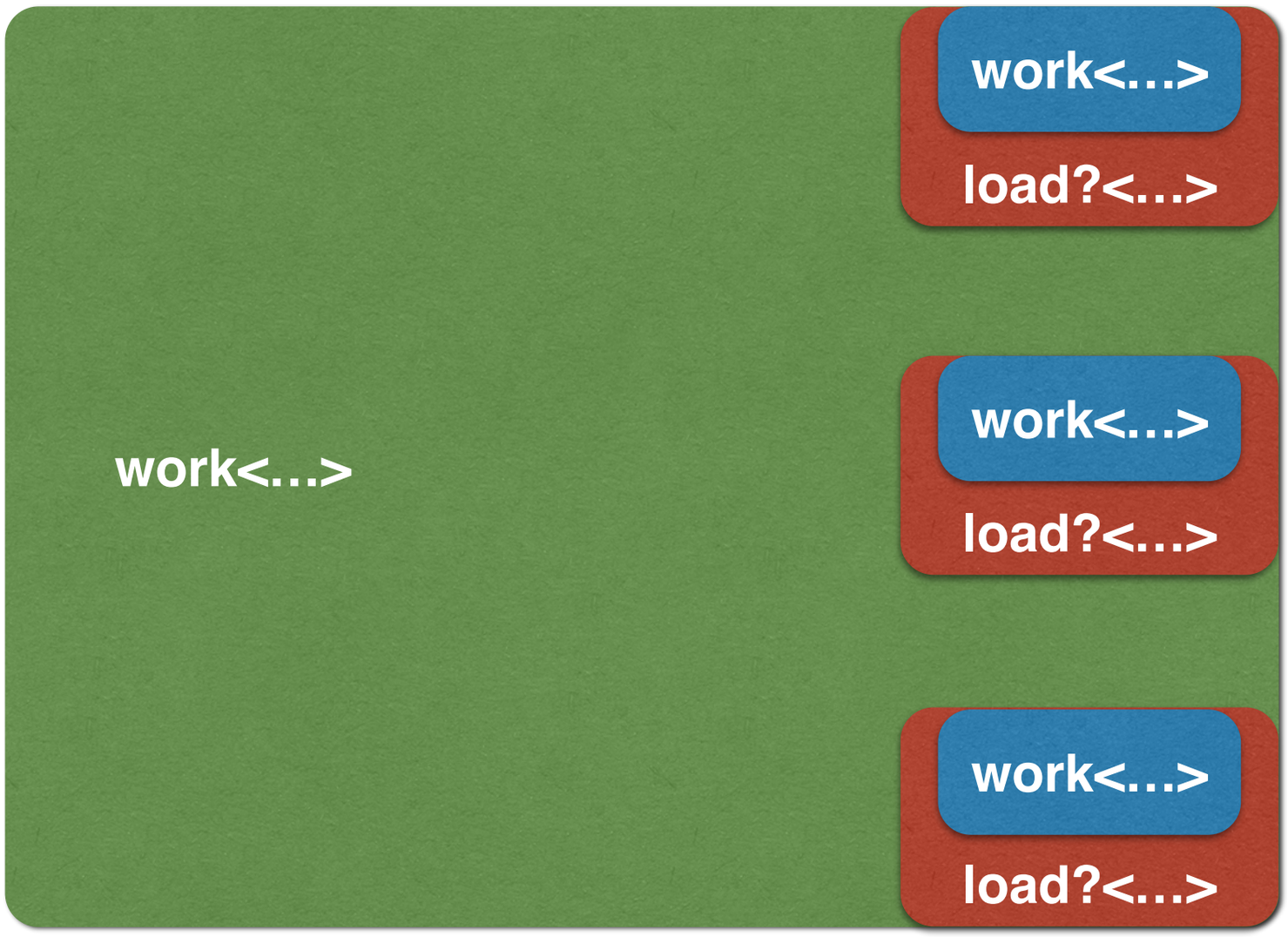






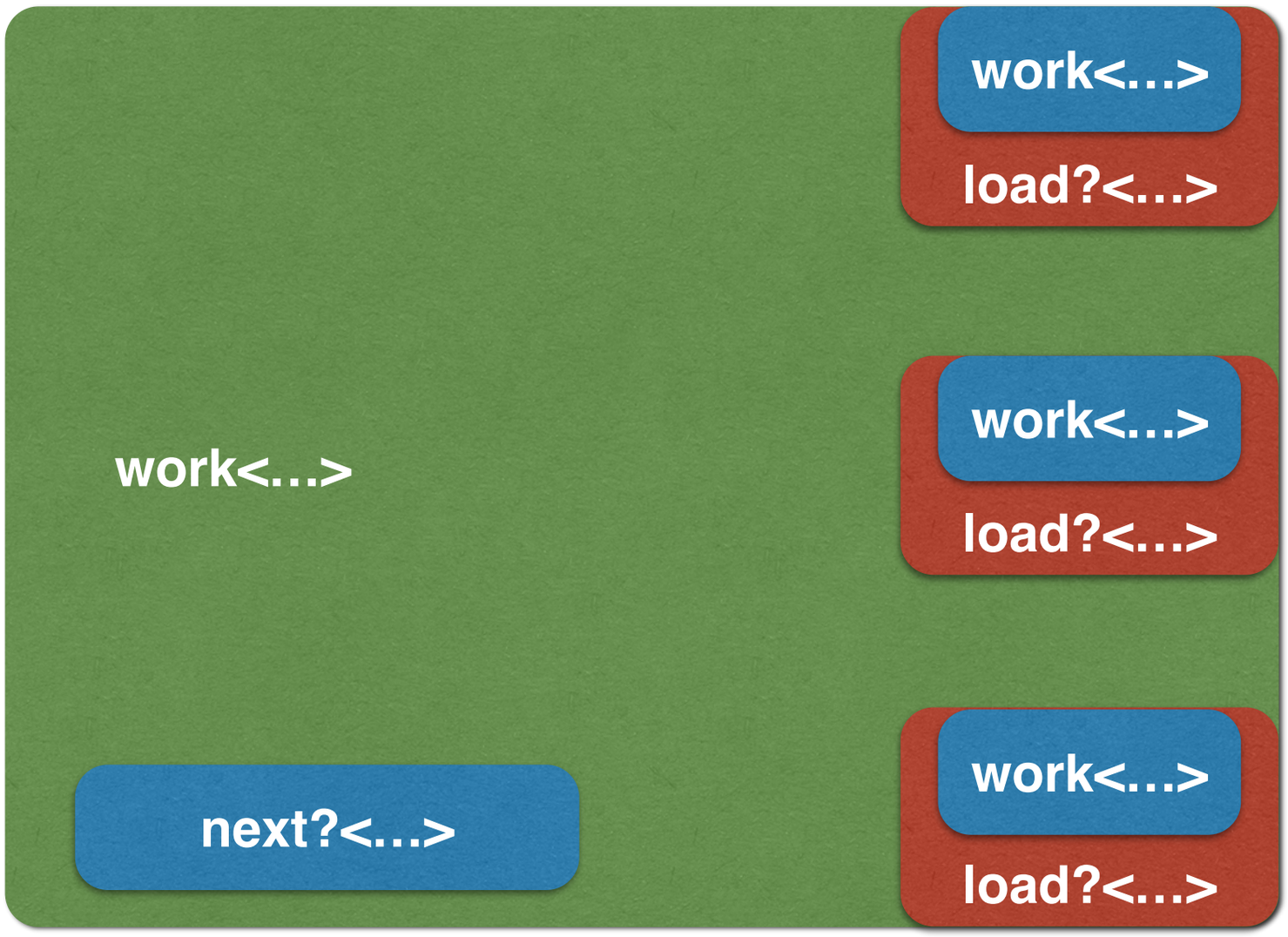
Server & Service Combinators

Load Balancing



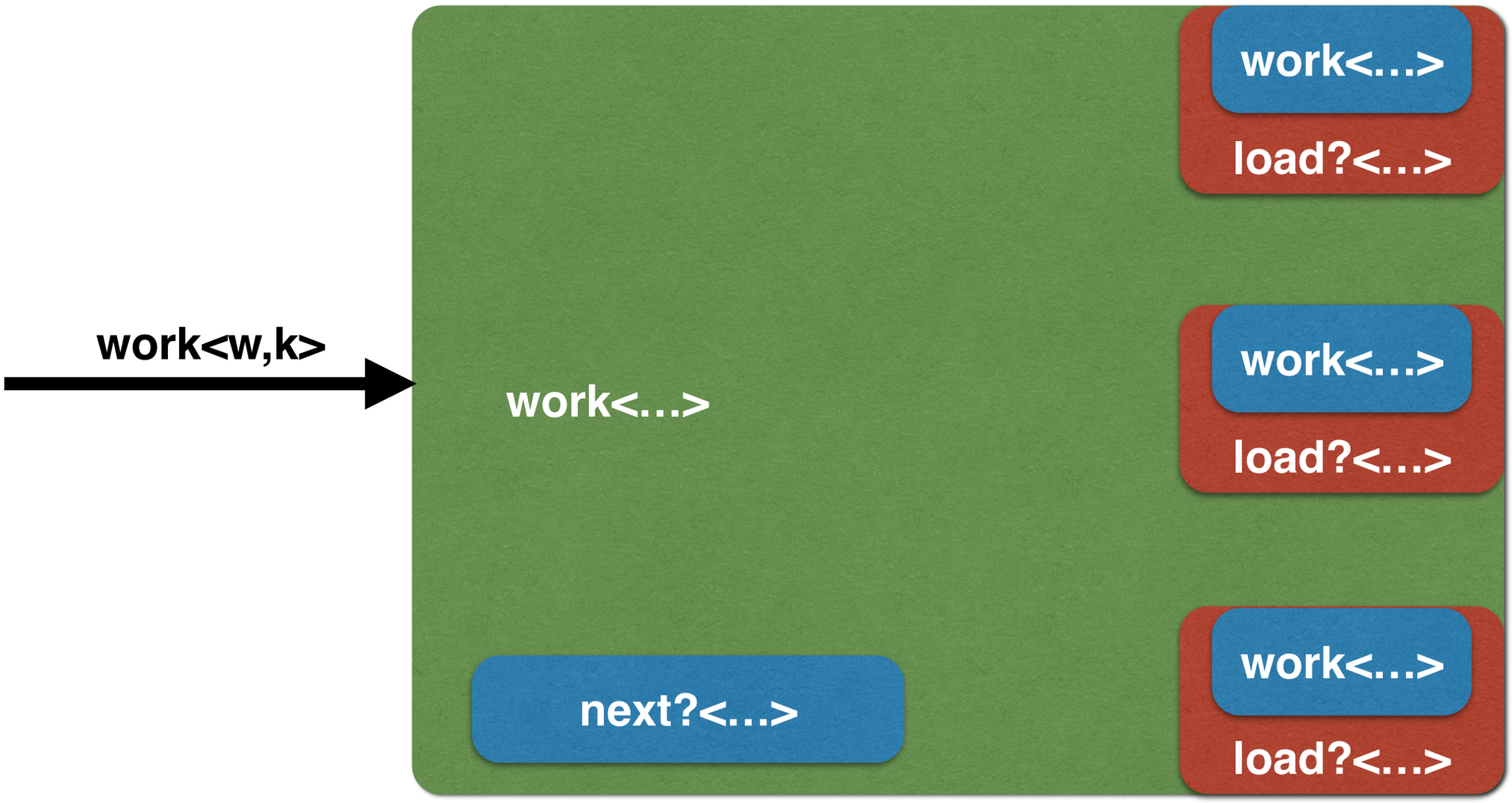
Server & Service Combinators

Load Balancing



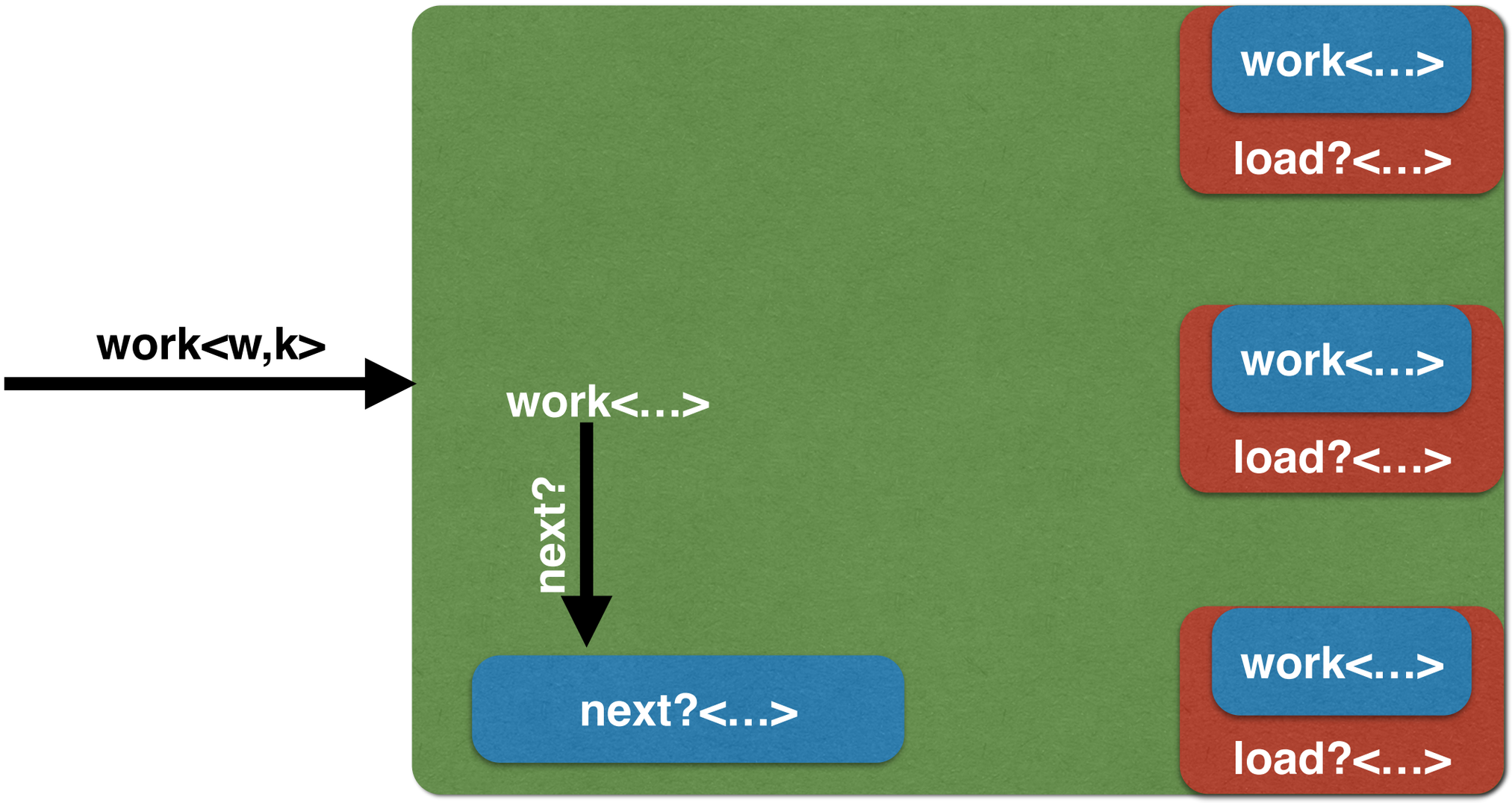
Server & Service Combinators

Load Balancing



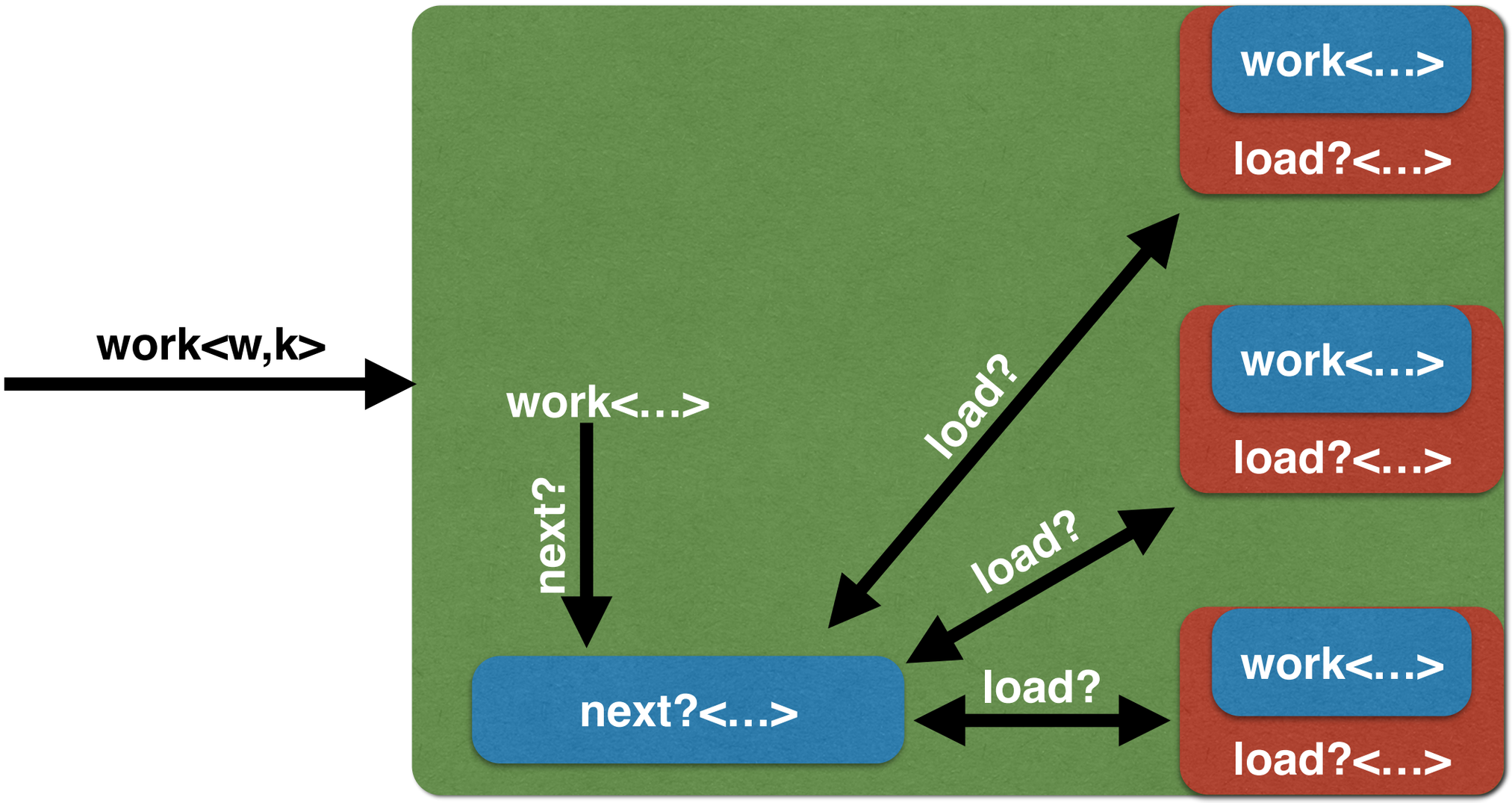
Server & Service Combinators

Load Balancing



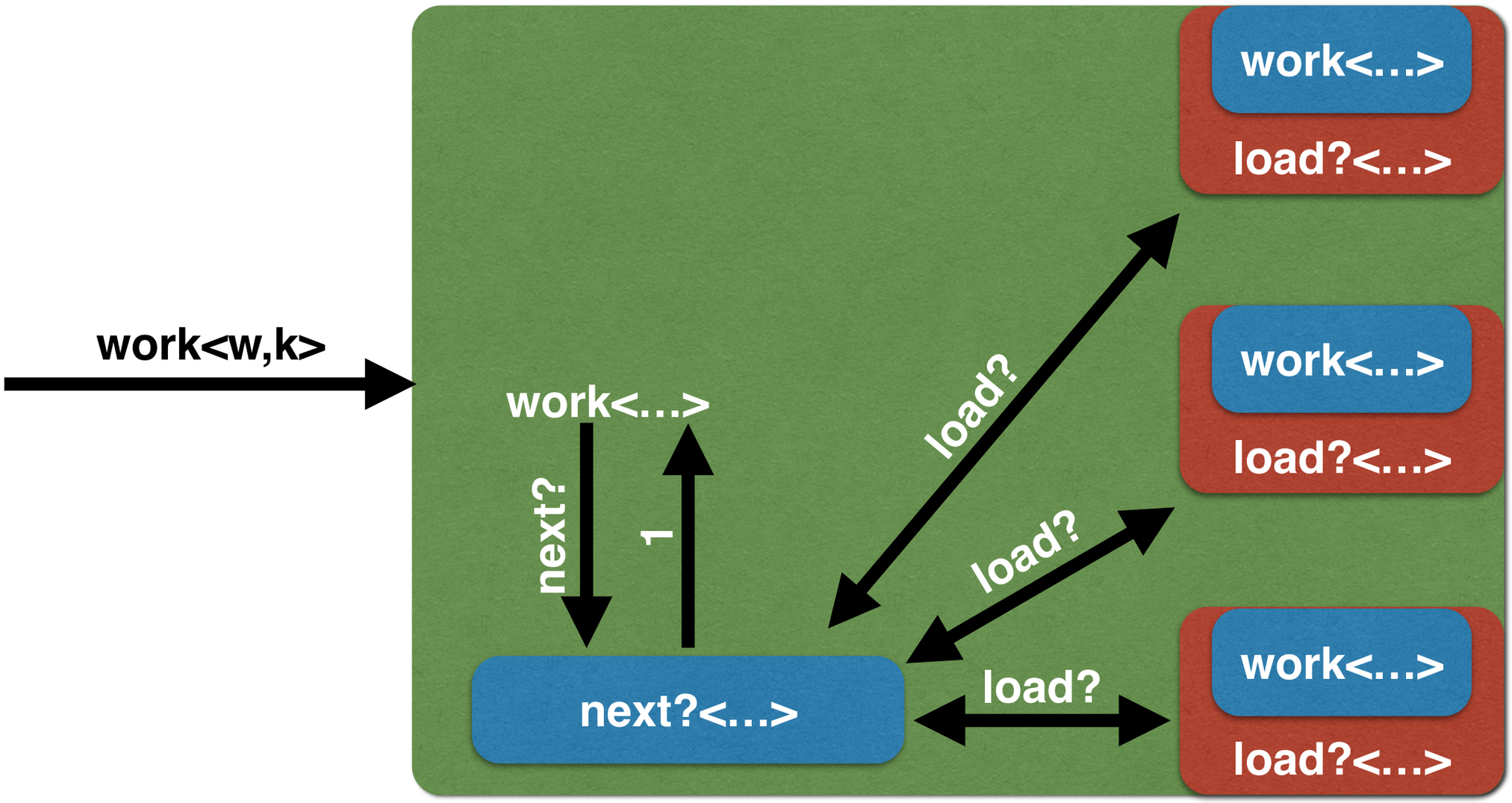
Server & Service Combinators

Load Balancing



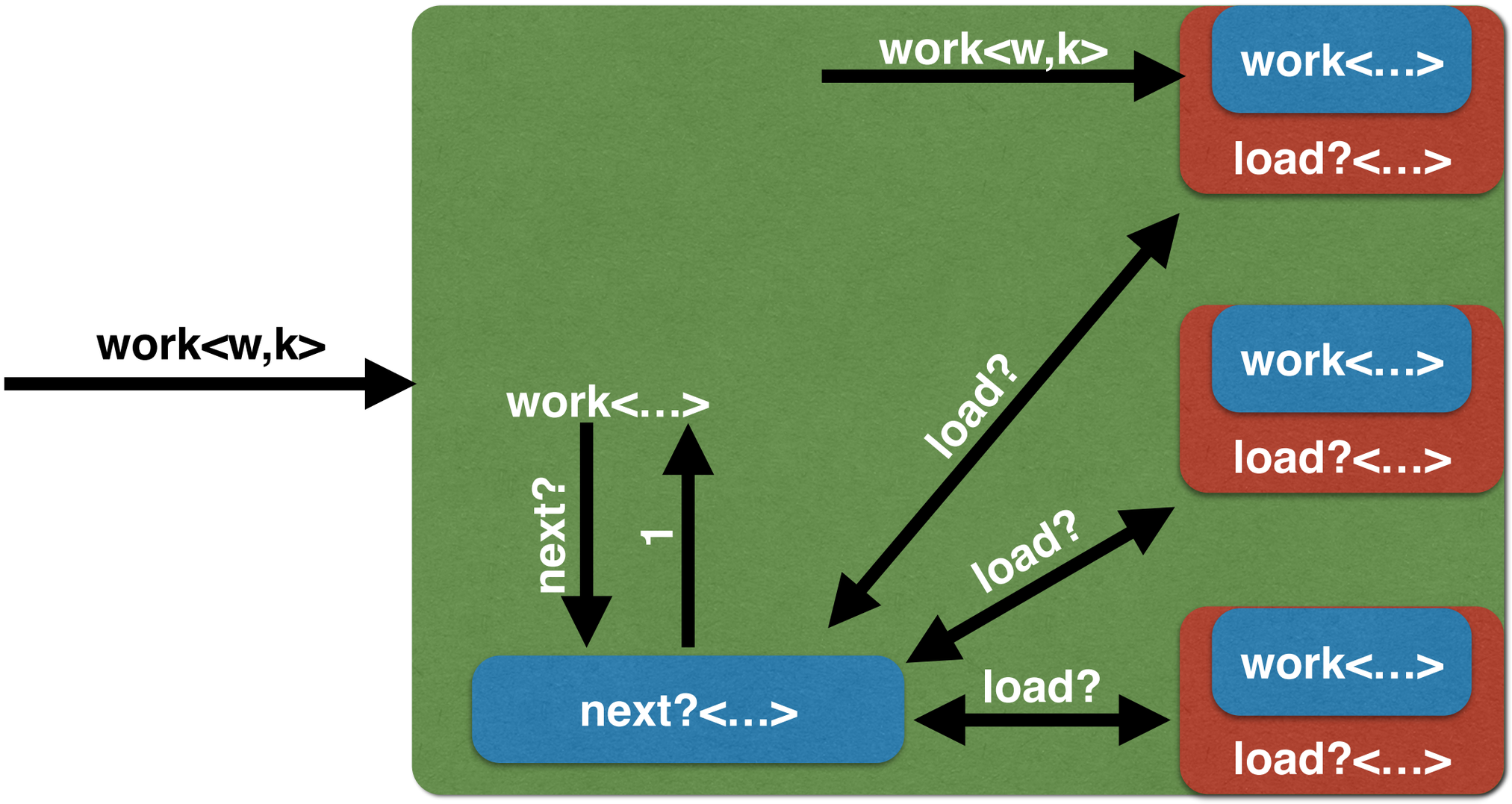
Server & Service Combinators

Load Balancing



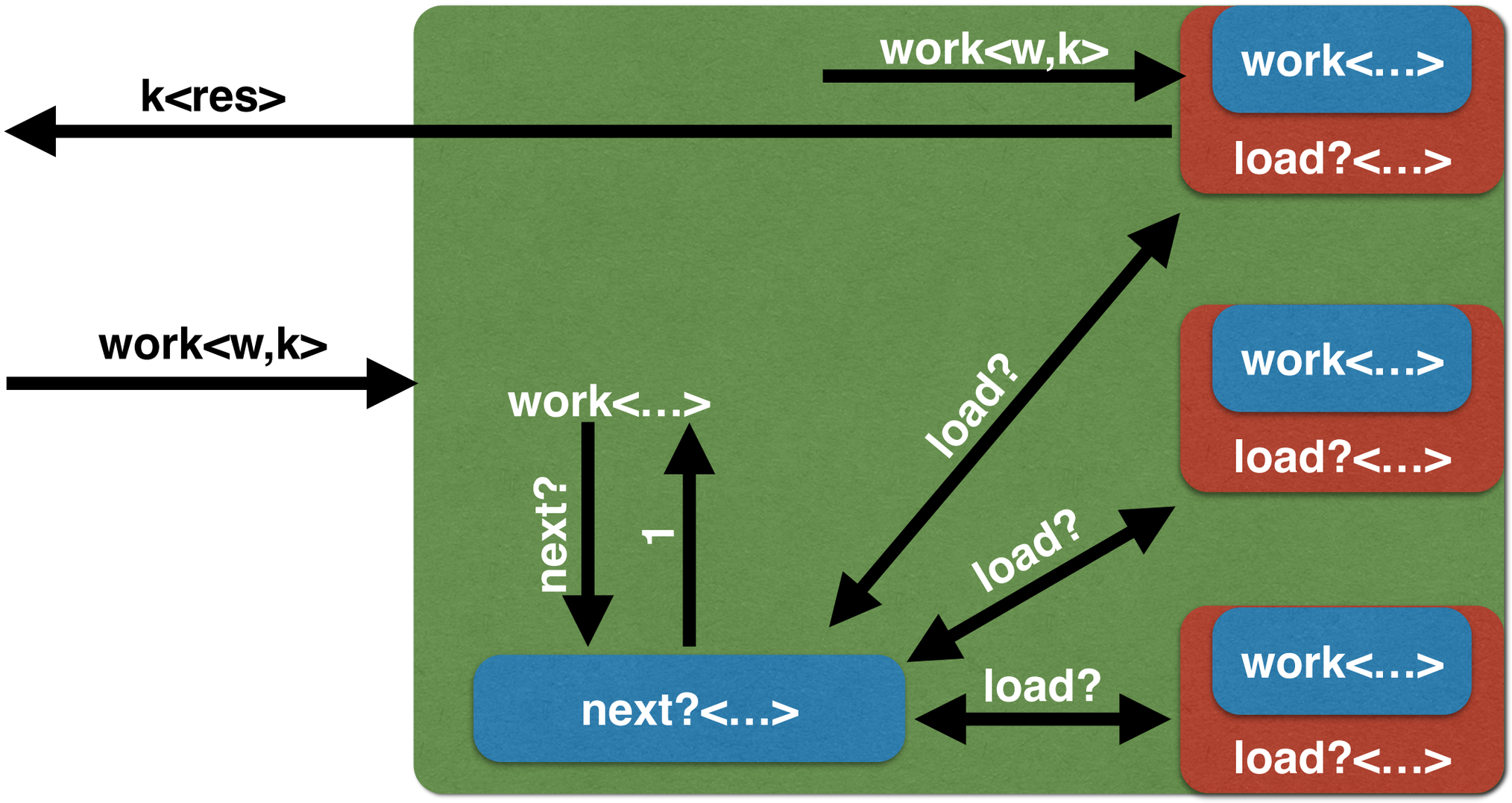
Server & Service Combinators

Load Balancing

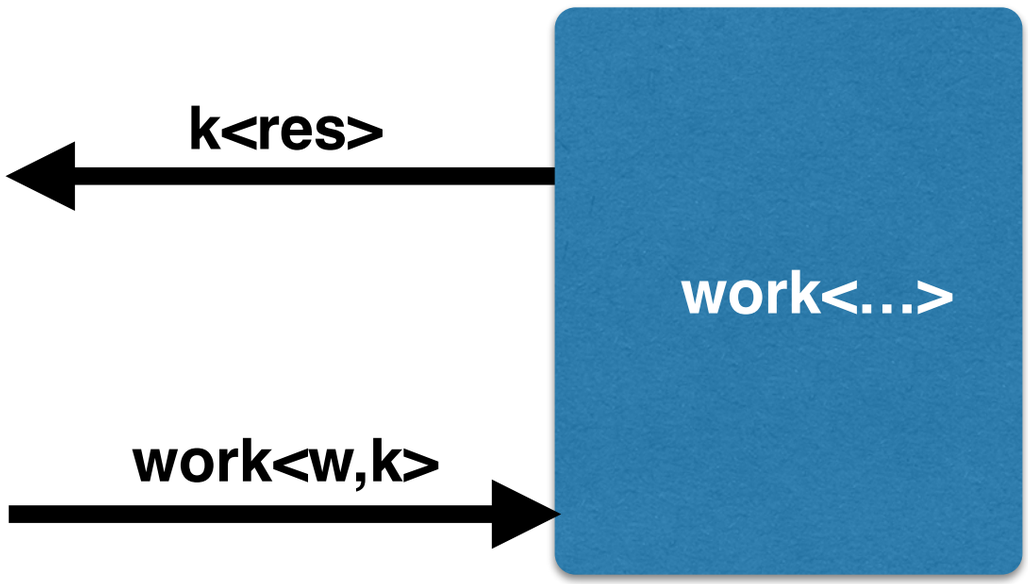


Server & Service Combinators

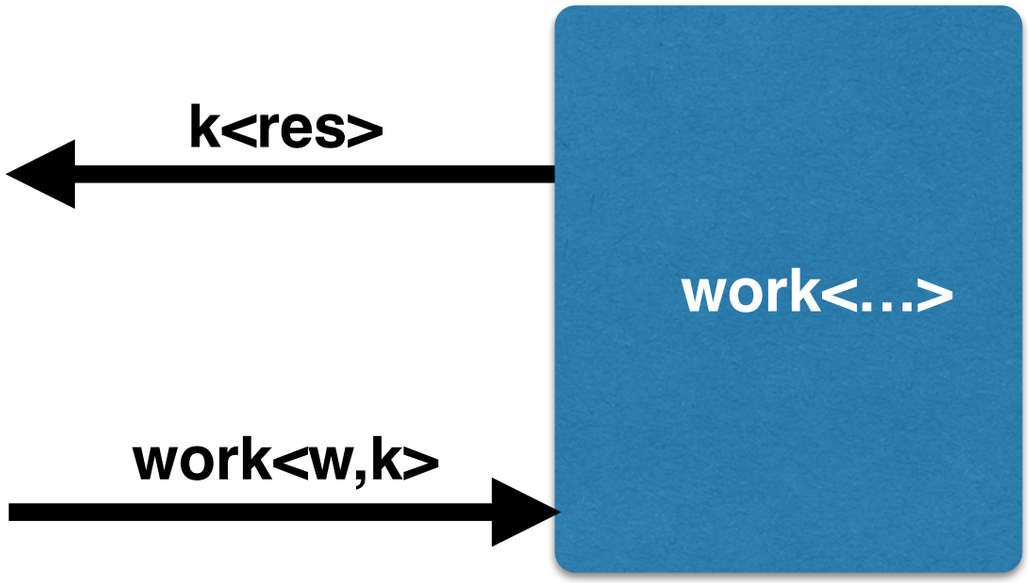
Load Balancing



Server & Service Combinators



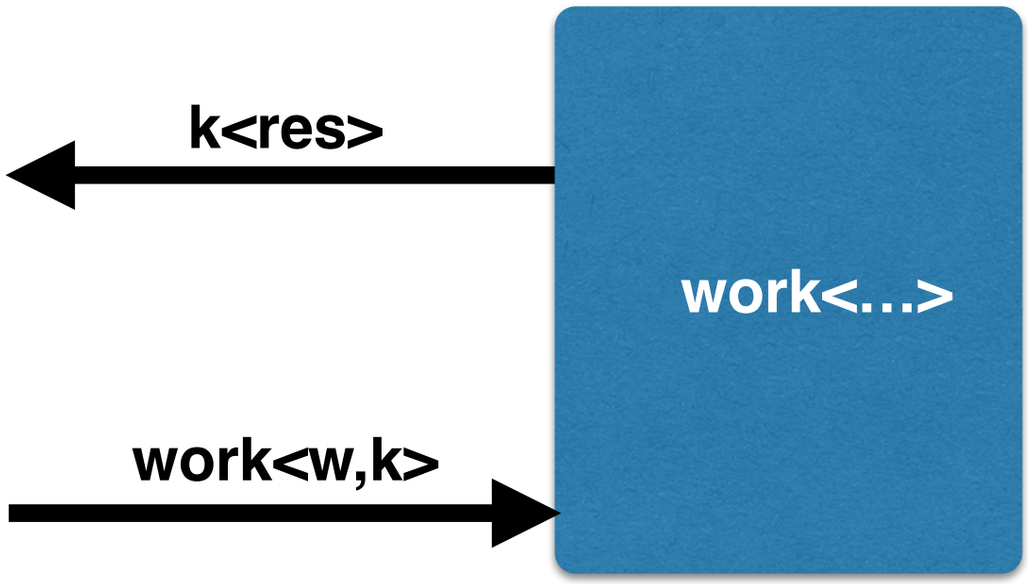
Server & Service Combinators



First class deployment pattern!

```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    } //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "      --stack ", { "Ref" : "AWS::StackName" },
                    "      --resource WebServer ",
                    "      --configsets wordpress_install ",
                    "      --region ", { "Ref" : "AWS::Region" },"\n"
        ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Server & Service Combinators



First class deployment pattern!

```
{ //...
  "Parameters" : {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [ "t2.micro", "t2.small",
      "ConstraintDescription" : "must be a valid EC2 instance type."
    } //...
  },
  "Resources" : {
    "WebServer": {
      "Type" : "AWS::EC2::Instance",
      "Properties": {
        "InstanceType"   : { "Ref" : "InstanceType" },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
                    "#!/bin/bash -xe\n",
                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v ",
                    "      --stack ", { "Ref" : "AWS::StackName" },
                    "      --resource WebServer ",
                    "      --configsets wordpress_install ",
                    "      --region ", { "Ref" : "AWS::Region" },"\n"
      ]]}}, //...
      },
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" :
[ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

Further combinators in the paper:

MapReduce Failure Recovery

Structural Type System

Based on System $F_{<:}$

Depth- & Width-Subtyping

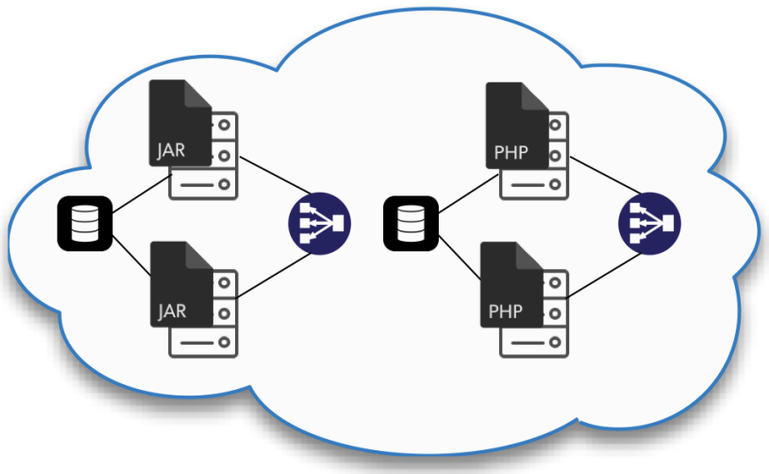
$$\texttt{Fib2} :: \mathbf{srv}\ \{\ \texttt{fib}\colon \langle \mathsf{Int}, \langle \mathsf{Int} \rangle \rangle\ \}$$
$$\texttt{Fib} :: \mathbf{srv}\ \{\ \texttt{fib}\colon \langle \mathsf{Int}, \langle \mathsf{Int} \rangle \rangle, \texttt{acc}\colon \langle \mathsf{Int}, \mathsf{Int}, \mathsf{Int}, \langle \mathsf{Int} \rangle \rangle\ \}$$

$$\texttt{Fib} <: \texttt{Fib2}$$

Bounded Polymorphism

$$\texttt{MKBalanced} :: \forall\alpha.\forall\omega <: \mathsf{TLAWorker}[\alpha].\ \mathbf{srv}\ \{\ \texttt{make}\colon (\mathsf{List}[\omega], \mathsf{Choose}[\omega]) \to \mathsf{TWorker}[\alpha]\ \}$$

Service Requests “Can’t Go Wrong”: Soundness Proof in Techreport



Summary & Outlook

Cloud Deployment DSLs

- Inextensible middleware
- Two-phase staging
- Unsafe parameterization

CPL

- Middleware combinators
- Dynamic Deployment
- First-class servers, CPS, Types

Paper

- Formal syntax & semantics
- Combinators for load balancing, fault tolerance, MapReduce
- Concurrent, single machine implementation in Scala

Ongoing Work

- Distributed, cloud-based implementation

λ Calculus Encoding

$$\lambda x.\ e \rightsquigarrow \mathbf{spwn}\ (\mathbf{srv}\ \mathtt{app}\langle x, k\rangle \triangleright \mathsf{T}(e, k)), \quad \text{where } k \text{ is fresh}$$

$$\mathsf{T}(\lambda x.\ e, k) = k\langle \mathbf{spwn}\ (\mathbf{srv}\ \mathtt{app}\langle x, k\rangle \triangleright \mathsf{T}(e, k))\rangle$$

$$\begin{aligned}
\mathsf{T}((f\ e), k) = \mathsf{T}(f, (\mathbf{spwn}\ (\mathbf{srv}\ \mathtt{k_1}\langle v_f\rangle \triangleright \quad & \text{where } v_f \text{ is fresh} \\
\mathsf{T}(e, (\mathbf{spwn}\ (\mathbf{srv}\ \mathtt{k_2}\langle v_e\rangle \triangleright \quad & \text{where } v_e \text{ is fresh} \\
v_f \sharp \mathtt{app}\langle v_e, k\rangle))\sharp \mathtt{k_2})))\sharp \mathtt{k_1}) &
\end{aligned}$$

$$\mathsf{T}(e, k) \quad = k\langle e\rangle$$

[Fournet and Gonthier '96]

Derived Syntax

$$\textbf{let}\ x = e_1\ \textbf{in}\ e_2 \quad \rightsquigarrow \quad (\textbf{spwn}\ (\textbf{srv}\ \texttt{let}\langle x\rangle \triangleright e_2))\sharp\texttt{let}\langle e_1\rangle$$

$$\textbf{letk}\ x = e_1\langle \overline{e}\rangle\ \textbf{in}\ e_2 \rightsquigarrow e_1\langle \overline{e}, (\textbf{spwn}\ (\textbf{srv}\ \texttt{k}\langle x\rangle \triangleright e_2))\sharp\texttt{k}\rangle$$

$$\textbf{thunk}\ \ e \rightsquigarrow \textbf{srv}\ \texttt{force}\langle k\rangle \triangleright k\langle e\rangle$$