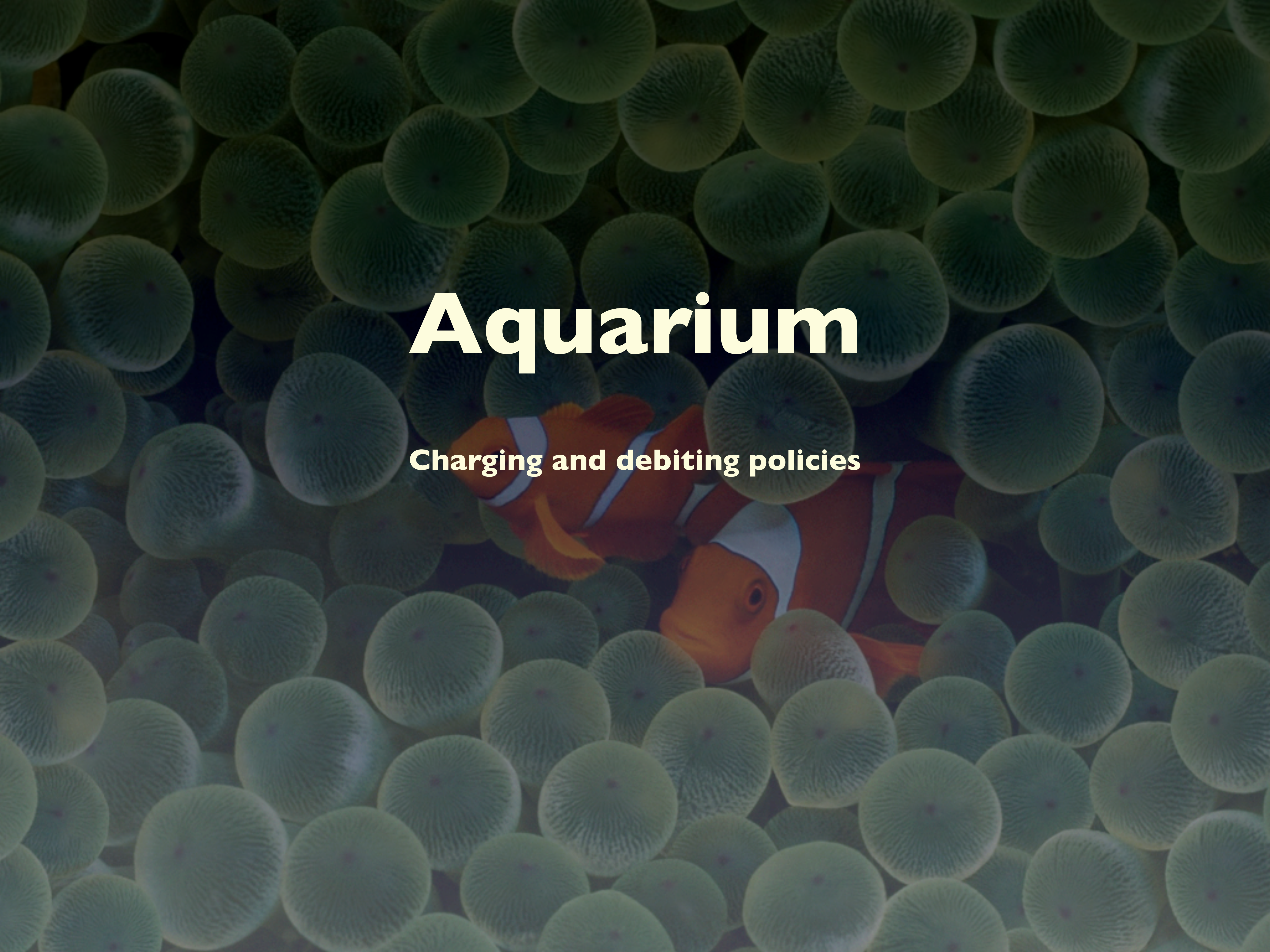


Aquarium

Charging and debiting policies

Charging

- Puts a price on resource usage
- Based on events
  - Continuous
  - Descrete
- Entities
  - Pricelists
  - Policies
  - Agreements
  - Time

Resources

- Time (VM flavor, software license)
- Space (disk, bandwidth)

```
resources:
  - bandwidthup
  - bandwidthdown
  - vmtime
  - diskspace
```

Pricelists

Prices for each resource over time

```
pricelists:
    - pricelist: &defpricelist
      name: default
      bandwidthup: 0.01
      bandwidthdown: 0.02
      vmtime: 0.1
      diskspace: 0.05
      effective:
        from: 0
   - pricelist: &other
     [...]
```

Policies

Algorithms for calculating prices over time

```
policies:
    - policy: &defpolicy
      name: default
      bandwidthup: $price times $volume
      bandwidthdown: $price times $volume
      vmtime: $price times $volume
      diskspace: |
                if $volume lt 100 then
                  $volume times $price
                elsif $volume gt 100 and volume lt 300 then
                  $volume times $price times 1.2
                else
                  $volume times price times 1.4
                end
      effective:
        from: 0
```

Agreements

Composition of policies with pricelists over time

```
agreements:
    - agreement:
      name: default
      policy: defpolicy
      pricelist: defpricelist
      effective:
        from: 0
```

Each “user” is assigned an agreement

Effectivity periods

```
effective:
  from: 0
  to: 124443
  repeat:
    - start: "30 12 * * Mon-Fri"
      end:   "00 14 * * Mon-Fri"
    - start: "00 18 * * *"
      end:   "00 20 * * *"
```

Inheritance

Facilitates the definition of specialized policies, pricelists or agreements, while retaining default values

```
- policy: &defpolicy
  name: default
  bandwidthup: $price times $volume
  bandwidthdown: $price times $volume
  vmtime: $price times $volume
  diskspace: $price times $volume
  effective:
    from: 0
```

```
- policy: &expbandwidth
  name: moreexpensivebandwidth
  extends: default
  bandwidthup: $price times $volume times 1.5
  effective:
    from: 0
    repeat:
      start: “0 12 * * Mon-Fri”
      end:   “0 18 * * Mon-Fri”
```

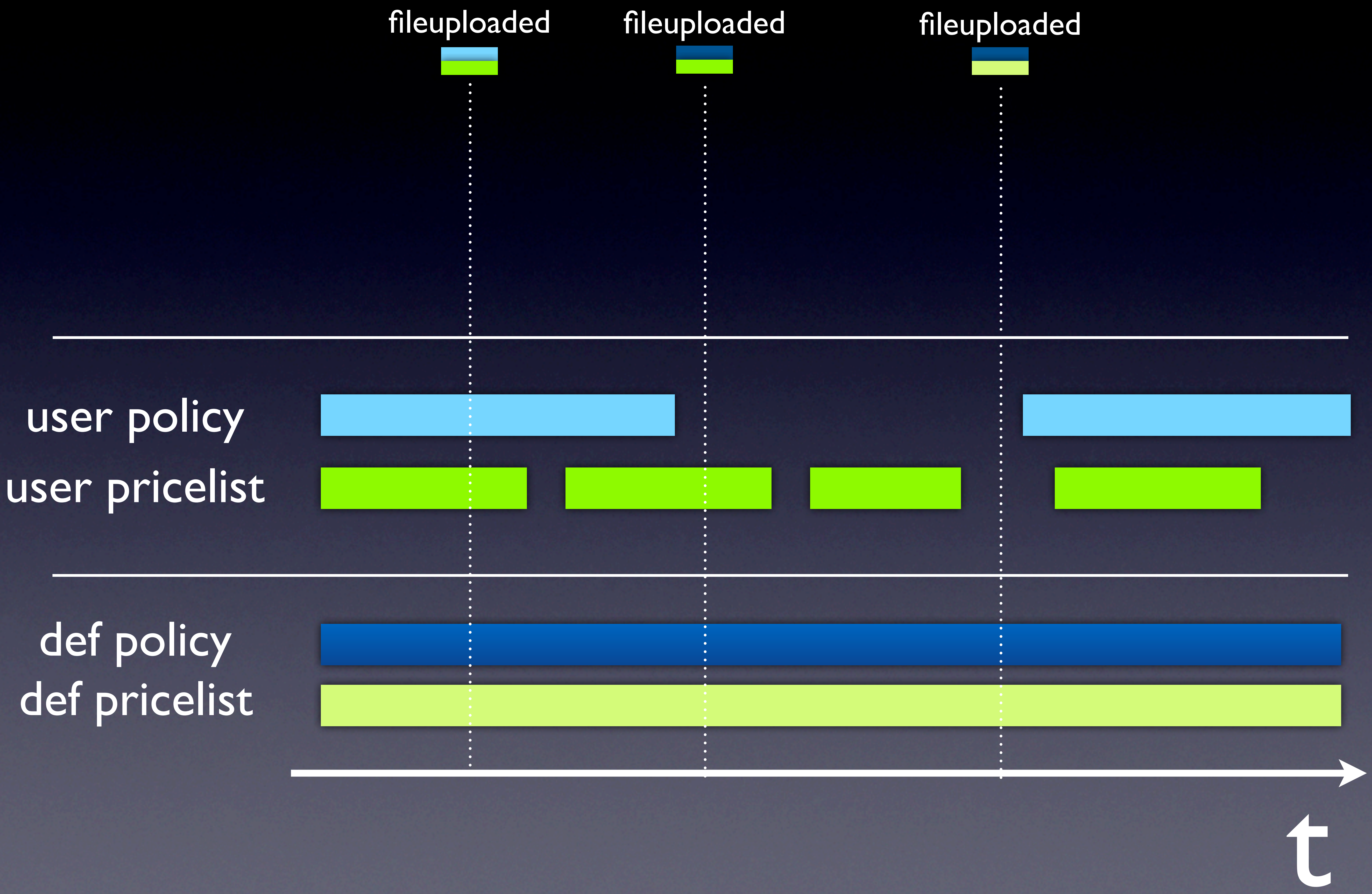
Events

- Represent a state change on a resource
- Persisted on an immutable log

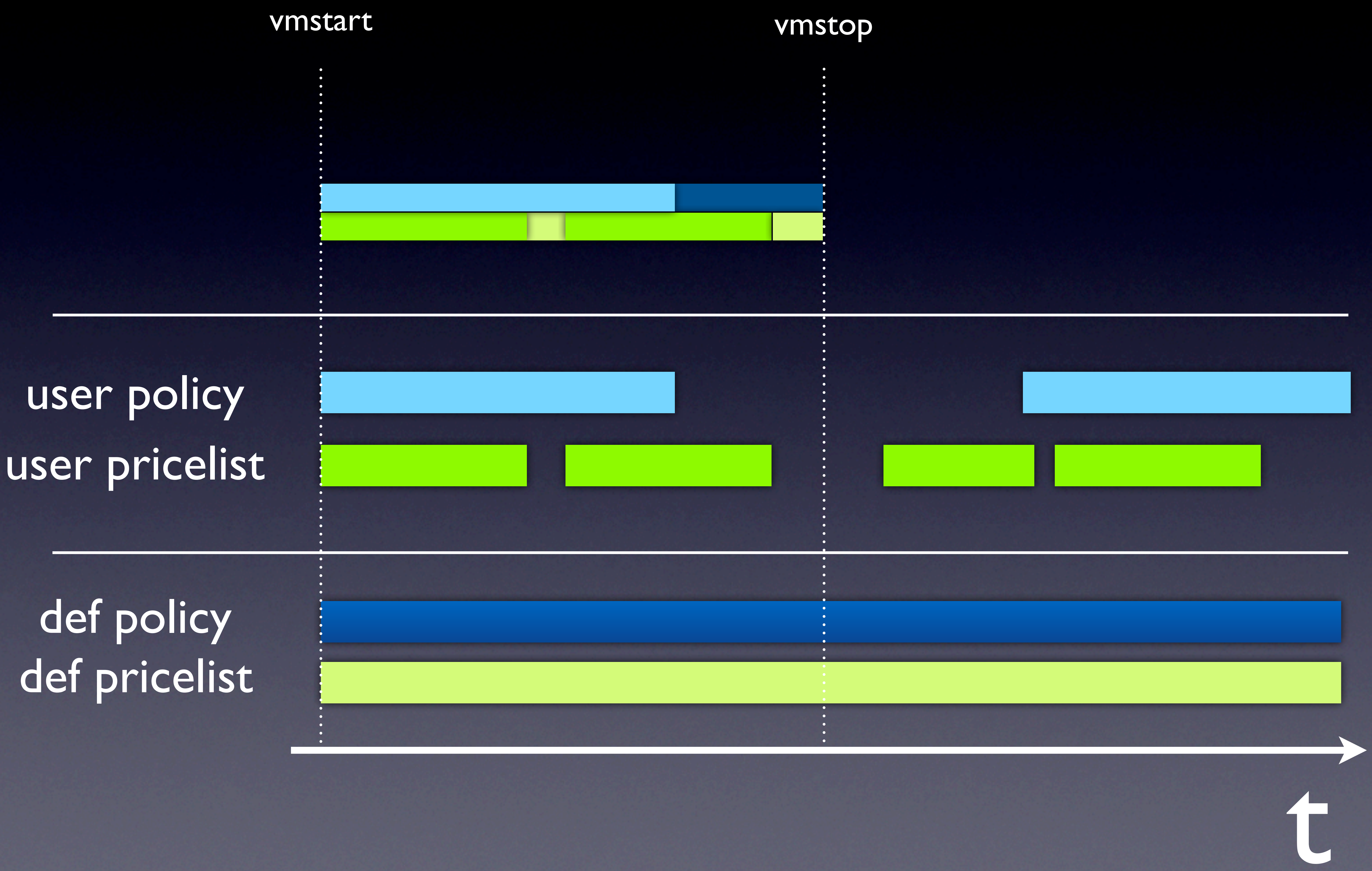
```
type: “fileuploaded”
id: 124443
when: “2011-10-30 12:32:33”
who: 12
bytes: 1443332
```

```
type: “vmstopped”
id: 124443
when: “2011-10-30 12:32:33”
who: 12
vmid: 14
```

Charging (descrete event)



Charging (continuous event)



Credit-related concepts

- Credit type
- Grouping
- Credit holder
- Credit origin
- Credit distribution policy
  - When
  - How

Credit type

- No specific type (!)
- Just an integer (double?)
- Support for positive & negative credits
- Negative credits better than shutting service down (?)

Credit grouping

- Support ‘credit structures’
  - Lab, Department, University
  - Division, Business Unit, Company
  - Not necessarily organizational structures but could be similar/the same
- Initiated by an appropriate user
- Contain other groups or plain users

Credit holders

- One of two types
  - Simple holder (user)
  - Composite holder (group)
- Wallet (?)

Credit origin

- Where did these credits come from?
  - Owned by a user directly
  - Inherited from (distributed by) a group

Credit distribution policy (How)

- FixedAny
- FixedEqual
- OnDemandUnlimited
- OnDemandMax
- Algorithmic
  - 0.80 * distributed_credits / members

Credit distribution policy (When)

- Manual
  - Group owner decides when
- Periodic
  - A ‘credit cycle’ modeled (or not) after the ‘billing cycle’ ?
- On credit ‘arrival’

DSL for groups

```
credit-group:
  - name: Lab
  - uri_label: lab
  - owner: some:uri # another group or user
  - members:
    - memberURI_1 # Can be teaching assistant
    - memberURI_2 # Can be a student
    - memberURI_3
  - credit_distribution:
    - when: OnCreditArrival
    - how: FixedEqual
```