Scheduling Instances
Scheduling individual instances

Info

It should be noted that preferring specific hosts is not always advantageous and can adversely a ect the uptime of your instances by denying them access to compute hosts. Rules forcing your instances onto only one or two hosts can result in the Instances not being scheduled if those hosts are already busy. Use these scheduling rules with caution for these reasons.

Scheduler requirements expressions

Scheduling an individual instance or template to a host can be done by adding requirements to the template or instance.

To schedule to a instance to a host:

```
SCHED_REQUIREMENTS = "<EXPRESSION>"
```

Likewise, scheduling an instance to a datastore can also be done. This is only possible when there are multiple datastores available, such as clusters with both HDD and SSD datastores attached.

To schedule an instance to a datastore:

```
SCHED_DS_REQUIREMENTS = "<EXPRESSION>"
```

Network interfaces can also be scheduled to pick appropriate virtual networks. For instance, if you set labels for the security levels of your networks:

```
NIC = [ NETWORK_MODE = "auto",
        SCHED_REQUIREMENTS = "SECURITY_ZONE = \"dev\"" ]
```

An expression is a combination of VARIABL relating to the compute node, network or datastore, along with an operator and a value to compare against.

Operator	meaning
=	equal to
!=	not equal to
>	greater than
<	less than

Operator	meaning
`@>`	array contains

Values can be strings, numbers or arrays of values.

Expressions can be joined together:

form	meaning
`EXPRESSION1 & EXPRESSION2`	boolean AND
`EXPRESSION1 | EXPRESSION2`	boolean OR
`! EXPRESSION1`	boolean NOT
`( EXPRESSION1 )`	nested sub-expression

Example

Schedule on a compute host with at least 3 idle cores and with more than 2.2 GHz CPU speed:

```
SCHED_REQUIREMENTS = "FREE_CPU > 300 & CPUSPEED > 2200"
```

Scheduler ranking

Scheduling can also be done by picking the top host in a list sorted by a numeric value. The variable used for this numeric ranking is speci ed in the `SCHED_RA` attribute.

For example, to schedule the instance against the compute host with the most available memory:

```
SCHED_RANK = "FREE_MEMORY"
```

Values in the ranking can be calculated from multiple variables using arithmetic operations addition `+`, subtraction `-`, division `/` and multiplication `*`. The compute host with the highest value is selected. Rankings can be inverted by placing a `-` in front of the variable.

Scheduling and ranking for compute hosts

For example, to prefer compute hosts with the slowest CPU speed:

```
SCHED_RANK = "-CPUSPEED"
```

Or to prefer compute hosts with a higher percentage of free memory instead of absolute value of free memory:

```
SCHED_RANK = "FREE_MEMORY / ( MAX_MEM * 1024 * 1024)
```

Rankings and expressions can be used together in an instance to select a group of hosts, then pick the preferred host from that group.

For example, to pick the slowest ARM architecture host in a group for a low-priority task:

```
SCHED_RANK = "-CPUSPEED"
SCHED_REQUIREMENTS = "ARCH = \"aarch64\""
```

Host variables can be chosen from the following list:

Variable	Description
ARCH	Architecture of the host CPUs, x86_64 or aarch64.
MODELNAM	Model name of the CPU, e.g. AMD EPYC 3251 8-Core Processor
CPUSPEED	Speed in MHz of the host CPUs.
HOSTNAM	Name of host as shown in the dashboard.
MAX_CPU	Total CPU shares, equal to number of CPU cores x 100
MAX_MEM	Total memory in GB
USED_CPU	Used CPU shares - % CPU used x number of cores
USED_MEMORY	Used memory in KB
FREE_CPU	Available CPU shares - % CPU idle x number of cores
FREE_MEMORY	Available memory in KB
CPU_USAGE	Total CPU allocated including unused portions
MEM_USAGE	Total memory allocated including unused portions
NETRX	Network traffic received in bytes
NETTX	Network traffic transmitted in bytes

Scheduling and ranking for datastores

You may also choose to select and sort datastores in a similar manner to compute hosts. To allow for future VM expansion, select datastores with at least 100 GB free space:

```
DS_SCHED_REQUIREMENTS = "FREE_MB > 102400"
```

Or, to rank datastores by the used capacity:

```
DS_SCHED_RANK = "USED_MD"
```

Datastore variables can be chosen from the following list:

Variable	Description
NAME	Datastore name - rst datastore is called "system"
TOTAL_MB	Total capacity in MB
FREE_MB	Free capacity in MB
USED_MB	Used capacity in MB

Scheduling and ranking for networks

You may wish to select from a group of networks and ll up the remaining leases before moving on to the next group:

```
NIC = [ NETWORK_MODE = "auto",
        SCHED_REQUIREMENTS = "SECURITY_ZONE = \"qa\"",
        SCHED_RANK = "-USED_LEASES" ]
```

Network variables can be chosen from the following list:

Variable	Description
NAME	Virtual network name
USED_LEASES	IP address leases consumed on this network
VLAN_ID	VLAN tag of tra c on this network

Scheduling groups of instances

Info

Avoid pinning workloads to specific hosts as this reduces the scheduling options for your workload and may impact the uptime of your application. In general consider using instance anti-affinity to separate workloads vital to the business and avoid impacts from power or network link failures. Instance affinity can be useful to group workloads requiring very low latency communications but is usually not required.

Groups of instances in [Product name removed] can be scheduled relative to each other using VM Groups. This means that scheduling can either bind the instances to the same hosts with **affinity** or force the instances onto different hosts with **anti-affinity**.

To create the policy we use the VM Group object. VM Groups consist of roles, which come in four different types:

Instance affinity

Instances with role types will be placed on **the same** compute hosts.

```
ROLE = [
    NAME = "co-located-on-same-hosts",
    POLICY = "AFFINED"
]
```

Instance anti-affinity

Instances with matching role types will be placed on **different** compute hosts.

```
ROLE = [
    NAME = "spread-across-hosts",
    POLICY = "ANTI-AFFINED"
]
```

Host affinity

Instances with matching role types will be placed **on a specific set** of compute hosts.

```
ROLE = [
    NAME = "only-use-these-hosts",
    HOST_AFFINED = "4,5,6"
]
```

Host anti-affinity

Instances with matching role types will be placed **to avoid a set** of compute hosts.

```
ROLE = [
    NAME = "avoid-these-hosts",
    HOST_AFFINED = "1,2"
]
```

Group to Group affinity

Groups with matching role names will be placed **on the same** hosts.

```
AFFINED = "co-located-on-same-hosts, group2"
```

Group to Group anti-affinity

Groups with matching Role names will be placed **on different** hosts.

```
ANTI_AFFINED = "avoid-these-hosts, only-use-these-hosts"
```

Linking the VM Group policies to instances and templates

Once the VM Group is created, the instances and templates are linked to it using the VMGROU field.

```
VMGROUP = [ VMGROUP_NAME = "app group 1", ROLE = "database" ]
```

Custom variables in expressions and rankings

Any system label is usable for scheduling but, user-created labels can also be selected. An example use case for this might be to add a label to compute hosts according to the rack containing them. For instance, a RACK field could be added and each compute host given a value according to where it had been installed. Then, a workload could either be limited to a single rack to avoid network traffic between racks, or use anti-affinity spread a workload across several racks at once.