MASTER OF SCIENCE
IN ENGINEERING

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Information and Communication Technologies (ICT)

Dockerisation d'environement pour Les projets de bioinformatique

Author:
Déruaz Vincent

Under the direction of:
Prof. Carlos Andrés Pena
CI4CB at HEIG-VD

External expert:
[Title] [FirstName] [LastName]
Company/Lab

Lausanne, HES-SO//Master, February 5, 2017

Sometimes a scream is better than a thesis.
— Manfred Eigen

To my parents...

Acknowledgements

Cette thèse as àtà réalisé dans le cadre du projet Inphinity à l'HEIG-VD.

Elle fait suite à la thèse de master [TODO: TITLE] réalisée par [TODO: prenom+nom DIOGO].

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Key words:

Résumé

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Mots clés :

Contents

List of Figures

List of Tables

1 Introduction

Ce travail a été réalisé dans le cadre du projet INPHINITY [TODO: pour qui ?]. Avec l'émergence de bactéries résistantes aux antibiotiques devenant une problématique mondiale qui menace les progrès de la médecine moderne, une alternative prometteuse pour lutter contre des bactéries multirésistantes consiste à utiliser leurs prédateurs naturels, des bactériophages, virus mangeurs de bactéries. Ces bactériophages, inoffensif pour l'homme, sont extrêmement spécifiques, ne reconnaissant qu'un type bien précis de bactéries. Ceci présente l'avantage de ne pas détériorer la flore bactérienne humaine, mais pose, par contre, une limitation pour leur développement rapide. En effet, pour chaque type de bactérie il faut trouver le bactériophage correspondant. Face à la nécessité d'examiner systématiquement une multitude d'interactions possibles, le développement rapide des bactériophages comme alternative aux antibiotiques ne pourra se faire qu'avec l'aide d'un modèle permettant de prédire les interactions entre bactériophages et bactéries. Ceci permettra notamment de réduire le nombre de validations expérimentales nécessaires à l'identification du bactériophage approprié et contribuera à l'essor de cette voie thérapeutique.

Ce travail se place également dans la continuité d'une précédente thèse de master dont l'objectif était de prouver la pertinence d'une méthode d'analyse par *machine learning*. En effet, il s'agit d'une méthode permettant [TODO 1: compléter].

Dans la présente thèse, il est question de mettre en place plusieurs aspects permettant l'enrichissement du processus d'analyse de la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* .

Afin de réaliser ces objectifs, une première phase du travail a consisté à réaliser des états de l'art pour les différents domaines utilisés (cf.chapitre 2-Etats de l'art).

Plusieurs phases distinctes de travail ont été nécessaires durant ce travail.

Premièrement, il a fallu reprendre la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* et comprendre ce qu'il y a été fait. Les informations concernant la thèse de Mr.Leite Diogo nécessaires à la compréhension de ce travail ont été abordées dans l'introduction, pour davantage d'informations veuillez consulter la thèse en question.

Deuxièmement, une fois les objectifs de thèse fixés, il a été important de réaliser un état de l'art des différentes technologies et aspects techniques susceptibles d'être utilisés dans la présente thèse, voir chapitre chapitre 2-Etats de l'art .

Troisièmement, c'est uniquement après ces deux premières phases que le développement a pu commencer, voir chapitre chapitre 4-Parallelisation python3 et chapitre chapitre 5-Environnement et application. Durant cette phase, un certain nombre d'aspects ont été développés: Notamment, l'utilisation de python 3 afin de remplacer l'utilisation de python2, moins efficace.

De plus, on souhaite être capable d'automatiser le lancement de "l'application" et par la même occasion rendre le déploiement facile et unifié, quelle que soit la machine hôte, pour autant qu'elle utilise le système d'exploitation Linux. Ensuite, on souhaite pouvoir lancer l'analyse pour différentes configurations, créées à l'avance. Un autre objectif important était de remplacer l'utilisation d'une API en ligne par une utilisation de sa version locale cf.chapitre 6-Déploiement.

Finalement, le temps de travail étant limité, il faut penser aux utilisations futures de ce qui a été développé. Ceci passe notamment par l'utilisation de l'application réalisée dans ce travail de manière simple voir chapitre chapitre 7-Simplification d'usage, mais aussi par les améliorations possibles à cette thèse, voir chapitre chapitre 9-Améliorations. C'est pour cela qu'un environnement de développement et d'exécution Docker a été produit dans ce travail, qui pourra être utile à d'autres membres du projet.

Il faut aussi préciser que certains résultats et métriques ont été réalisés et sont regroupés dans le chapitre chapitre 8-Résultats et Benchmarks.

2 | Etats de l'art

2.1 introduction

Dans ce chapitre, nous aborderons les différentes pistes envisagées afin de remplir les objectifs fixés dans cette thèse, comme listés dans l'introduction (chapitre 1-Introduction).

Avant toutes choses, il a fallu se mettre au niveau et comprendre la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* .

2.2 Automatisation

En terme d'automatisation, une pratique bien courante chez les développeurs s'agis d'utiliser des scripts bash afin de pouvoir exécuter un certain nombre de commande et de code successivement. Bien que cette méthode présente l'avantage d'êtres simple, il suffit d'une console UNIX et d'un éditeur de texte, elle présente un défaut majeur. En effet, le développeur du script contrôle quel commande et code sont exécutés et peut également définir des paramètres pour ceux-ci, mais il ne peut pas contrôlé l'environnement d'exécution.

Une façon de faire, en plein essor depuis quelque temps, est l'utilisation de la plateforme Docker. Il s'agit d'un logiciel de containerisation. C'est-à-dire la création de brique d'application, qui mise en communes permet de réaliser une application globale. De plus, le développement d'une telle solution permet un partage facilité grâce à un déploiement facilité et autonome. Pour davantage d'explication sur le sujet je vous renvoie au chapitre chapitre 3-Bases de Docker.

Vous l'aurez bien compris, le choix qui a été fait est celui de l'utilisation de Docker.

2.3 Configuration

En ce qui concerne la recherche d'une méthode afin de réaliser facilement des fichiers de configurations précréées, beaucoup de solutions existent. Ces différentes méthodes sont plus ou moins flexibles aux modifications.

Les fichiers de configurations dont il est question ici sont spécifiques à la partie python du code qui sera exécuté par notre application chapitre 5-Environnement et application. En effet, l'on souhaite entre autres être capable de donner des fichiers de configuration en entrée et d'obtenir pour chacun un résultat en sortie.

Nous citerons ici uniquement la solution retenue, car les autres solutions trouvées sont soit trop incompatibles soit presque identiques à la solution retenue.

Nous utilisons le module python *Configparser*, qui permet de lire et parser des fichiers à l'extension .ini de manière simple. De plus, la structure d'un fichier .ini est très simple et ne laisse donc que très peu de place aux erreurs de format.

2.4 Hmmer

Dans la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* les séquences protéiniques sont recherchées dans la base de données de profile-HMM à l'aide d'une interface de programmation applicative (API) en ligne. Cette API est disponible depuis le site *https://www.ebi.ac.uk/Tools/hmmer/*.

Comme dis précédemment, un des objectifs de ce travail est de se passer de l'utilisation de cette API car sont accès n'est pas toujours disponible ou stable.

Une recherche rapide à permis de se rendre compte que l'application utilisée derrière cette API est disponible au téléchargement et peut donc êtres utilisée de manière locale. Pour davantage d'information chapitre 5-Environnement et application, sous-chapitre Hmmer.

2.5 Parallélisation

La version existante du code se trouvant dans la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* est une version sous forme de script, proof-of-concept, en python2 et non *multiprocessed*. Afin de garantir une utilisation optimale des ressources de la machine hôte, sur laquelle le code est exécuté, nous souhaitons rendre le code parallèle là où il est possible de le faire.

Plusieurs solution sont possible, encore une fois les solutions les plus compliquées ne sont pas toujours celle les plus efficaces. De plus une méthode trop complexe pourrait réduire la bonne transmission du code à d'autres développeurs.

La partie principale que l'on souhaite paralléliser est l'utilisation de la fonction de scanne de HMMMER, étant donné qu'un très grand nombre de séquences protéiniques doivent être analysées.

2.5.1 Simple

Docker

Docker, mis à part de rendre le déploiement et l'exécution d'application automatisée, permets également de lancer plusieurs conteneurs simultanément, chapitre 3-Bases de Docker. Un conteneur englobe un système de fichier complet possédant tous se qu'y est nécessaire a remplir sa fonction.

Python

En python on retrouve deux principales méthodes permettant de réaliser de code parallèle. En effet, on peut utiliser le *multiprocessing* ou le *multithreading*.

Notre bute est de réaliser et d'optimiser un code Central processing unit (CPU) dépendant, c'est-à-dire coeurs dépendants. Lors de l'utilisation du langage python il faut savoir qu'avec des codes CPU dépendants, python limite les possibilités de parallélisme à cause de la Global Interpreter Lock (GLI). La GLI est nécessaire en python, car python n'est pas *tread safe*. En effet, il y a, en python, un verrou global lorsque l'on essaye d'accéder à un objet depuis un thread.

À cause de se verrous les codes CPU dépendants ne gagneront pas en performance lorsqu'ils sont parallélisés à l'aide de *multithreading*, mais uniquement avec le *multiprocessing*.

2.5.2 Avancée

Docker Swarm

Une autre méthode utilisant une librairie avancée de Docker, consiste à utiliser Docker Swarm. Docker Swarm apporte à Docker une gestion native du *clustering*, afin de transformer un groupe de *Docker engines* en un unique et virtuel *Docker engine*. Grâce à cela, il est possible d'exécuter une application sur une architecture partagée sur plusieurs systèmes physiquement indépendants.

Spark

Spark est un framework *open source* de calcul distribué. Il permet d'effectuer des analyses complexes sur un grand nombre de données.

Il est également un ensemble d'outils pour le traitement de grande source de données, notamment grâce à des fonctions *MapReduce*.

2.6 Optimisations

Le code repris de la thèse *[Modélisation prédictive des interactions entre bactéries et virus bactériophages - Leite Diogo]* est un code séquentiel, sous forme de script nécessitant des inputs utilisateurs a chaque étape. De plus, ce code est ecrit en python dans sa version 2.

Grâce au travail du Dr. Brett Cannon, See here, on se rend compte que python 3.3 pourrait optimiser les performances de notre application. On peut lire ici que même l'appel des fonctions est en moyenne 1.20 fois plus rapide. De plus, les *threadded count* sont également plus rapide.

Une autre possibilité est d'utiliser *Cython*. Cython est un compilateur/langage de python permettant d'utiliser des appeles au langage C et de compiler un code python en exécutable C. Il faut savoir qu'un exécutable C est généralement plus rapide que l'exécution de l'interpréteur Python.

On trouve le tableau suivant dans la documentation de Cython, qui permet de nous rendre compte des différences.

Method	Time (ms)	Compared to Python	Compared to Numpy
Pure Python	183	x1	x0.03
Numpy	5.97	x31	x1
Naive Cython	7.76	x24	x0.8
Optimised Cython	2.18	x84	x2.7
Cython calling C	2.22	x82	x2.7

Figure 2.1 – Tableau de comparaison de Cython - `http://notes-on-cython.readthedocs.io/en/latest/std_dev.html/`).

2.7 conclusion

Après des tests sur ces différentes technologies et méthodes et quelques discutions ici et là, l'idée ayant été arrêté est d'utiliser *Docker* et *Docker Compose* comme contexte applicatif et de transformer les scripts en une application orientée objet en python 3.3 gérant les fichiers de configurations avec la librairie *Configparser*. Pour ce qui est du parallélisme, il sera réalisé en utilisant la librairie *Multiprocess*.

3 Bases de Docker

3.1 Introduction

Dans ce chapitre nous allons parler du fonctionnement de *Docker* et *Docker COmpose*. Ce chapitre est réalisé sous le ton d'un cours d'introduction à Docker, afin de pouvoir transmettre les connaissance de base a l'utilisation et à la modification du travail réalisé lors de cette thèse. Cela passera entre autre par certain exemple et code qui seront fournit en annexe notamment.

Docker permet l'execution de code dans un conteneurs indépendant de vote systeme hôte.

3.1.1 Utilisations

3.1.2 Compatibilité inter-OS

Docker permet d'éviter les problemes liés aux différences entre les environnements d'execution. En effet, lorsque l'on execute un code avec Docker ont contrôle exactement l'états et le type d'environnement d'execution. Cela rend donc possible l'execution d'un code sous différent Système d'explotation (OS) hôte (OSX, Linux, Windows).

Mais pourquoi ne pas utiliser une simple machine virtuelle ? Une première différence entre une machine virtuelle et DOcker est le fait que Docker n'encapsule pas tout un OS, ceci permet une execution beaucoup plus rapide, et c'est bien se que l'on cherche dans ce travail.

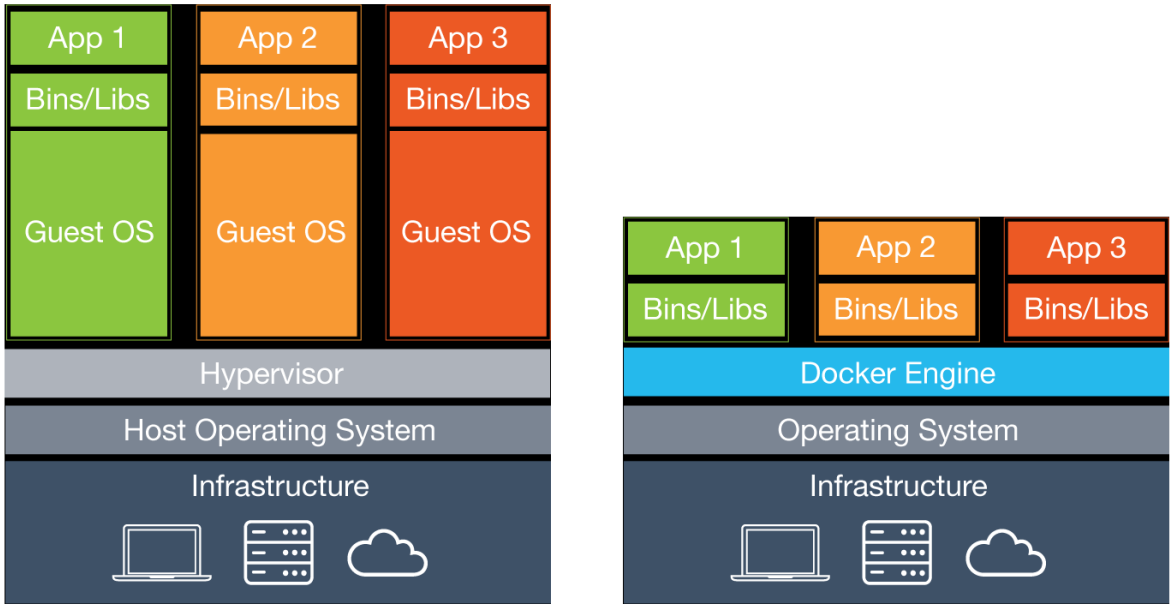

Figure 3.1 – Virtual machine vs Docker

3.1.3 Miracle or illusion

Je tiens ici à faire une mise en garde vis-à-vis de l'utilisation de Docker. Dans sont utilisation Docker est, à mon sens, une solution assez miraculeuse, notamment par le fait que l'on peu partage une application sans se poser de question sur l'hôte cible. Mais attention Docker n'est pas aussi miraculeux que cela dans le développement d'une solution applicative. En effet, il peu parfois etres compliquer d'arriver du premier coup à réaliser se que l'on souhaite.

Docker n'est donc pas une solution miracle, mais présente beaucoup d'avantages en terme d'execution standardisé, de partage de code et de déploiment.

3.2 Pré-requis

3.2.1 Connaissance

Il est necessaire d'etres à laise avec l'OS Linux et l'utilisation de commande UNIX. En effet, la pluspart du temps les conteneurs utiliserons un systeme Linux. Pour plus d'information cf. ci-après.

3.2.2 Installations

Installation Docker

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --
    ↪ recv-keys 58118E89F3A912897C070ADBF76221572C52609D

$ sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu
    ↪ -xenial main'
$ sudo apt-get update
$ sudo apt-cache policy docker-engine
$ sudo apt-get install -y docker-engine
$ sudo systemctl status docker
```

Vous devriez maintenant voir une sortie console semblable a celle de la figure 3.2:

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2016-11-30 01:28:41 CET; 8h ago
     Docs: https://docs.docker.com
 Main PID: 1171 (dockerd)
    Tasks: 41
   Memory: 3.7G
      CPU: 1min 18.316s
   CGroup: /system.slice/docker.service
           ├─ 1171 /usr/bin/dockerd -H fd://
           ├─ 1607 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --shim docker-con
           └─21678 docker-containerd-shim 1ed0c2cbc7dbd228de8ffd82a822df63abc6f75f3978b96a8b9e82832489a343 /var/run/d
```

Figure 3.2 – Services Docker opperationnel

Il faut à présent configurer Docker pour votre utilisateur hôte.

```
$ sudo usermod -aG docker $(whoami)
```

A se point ci, il vous faut redémarer votre machine.

Installation Docker-compose (1.9)

```
$ curl -L "https://github.com/docker/compose/releases/download/1.9.0/
    ↪ docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-
    ↪ compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ docker-compose --version
```

Vous devriez à présent obtenir la sortie console suivante:

```
docker-compose version: 1.9.0
```

3.2.3 Téléchargements

3.3 Fonctionnement

3.3.1 Docker

Il faut commencer par clarifier de quoi ont parle lorsque l'on utilise le mot conteneur. Il s'agit d'une *enveloppe* virtuelle permettant de packager une application ou un code avec toutes les dépendance necessaire au fonctionnement de l'application. On package donc les fichiers source, librairies, runtime, outils, fichiers, base de données, etc.

Un conteneur n'embarque pas de OS, il s'appuie sur celui de l'hôte sur lequel il est deployé. Ce qui rend un conteneurs baucoup plus légé qu'une machine virtuelle 3.1.

Il faut également spécifier que Docker opère une isolation, des conteneurs, au niveau du système d'explotation.

Un conteneurs DOcker est decrit à l'aide d'un simple fichier *.Dockerfile*, il décrit la création du conteneur, en détails. On peut personnaliser cette description de manière très détaillée.

Il faut voir une application réalisé avec Docker comme une somme de micro-services. Nous verrons dans la section suivante des exemples basiques d'application DOcker. Le but étant de:

- rendre l'application d'avantage élastique;
- Améliorer les performance;
- Le déploiement continue est facilité. On peu relancer les services independament les un des autres.

3.3.2 Docker-compose

Docker-compose permet de définir et d'executer des application multi-conteneurs. En effet, sans compose il fallait lancer les différent services de votre application soit manuellement soit en utilisant des scripts.

Compose utilise un fichier de composition, *docker-compose.yml*, afin de configurer une application Docker. Ce qui permet de lancer une application à l'aide d'une seule commande.

Pour résumer, une application Docker, utilisant Compose, est la combinaison de trois étapes:

1. Définir les différent Dockerfile des vos micro-services composant l'application;
2. Définir les services qui seront utilisé dans le compose, leurs relation et leurs configurations;
3. Lancer l'application avec la simple commande, *docker-compose up*.

3.4 Exemples

3.4.1 simple pull, build et run

Les trois commandes les plus importantes de Docker sont *pull*, *build* et *run*. En effet, ces commandes sont indispensable et doivent imperativement êtres comprise.

3.4.2 Serveur Web

3.4.3 Biopython

3.4.4 Parallélisation

3.5 Conclusion

3.5.1 Docker

3.5.2 Alternatives

4 Parallelisation python3

4.1 Code de base

4.2 Utilisation

5 Environnement et application

5.1 Images Docker

5.1.1 Hmmer

5.1.2 Database

5.1.3 Core

5.2 Docker Compose

5.3 «Inphinity»

6 Déploiement

6.1 Obtention des sources

6.2 •

7 Simplification d’usage

7.1 Commandes et alias

7.2 Scripts

8 Résultats et Benchmarks

8.1 Parallèlisation

8.2 Dockers

8.3 Phases

9 Améliorations

9.1 Parallèlisation

9.2 Machines Amazone

9.3 Spark

10 | Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

A | An appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.