1. **מערכת מונוליטית** - מערכת המאופיינת בכך שכל מרכיביה שזורים האחד בשני. מערכת כזו אינה מפרידה בין הלוגיקות השונות שלה. כל המערכת בנויה כמקשה אחת. במערכת כזו קשה לייצר מודולריות, קשה להבין את זרימת הפעולות. קוד של מערכת כזו בדרך כלל יהיה מאוד לא קריא. בנוסף, בכל פעם שעובדים על חלק (אפילו קטן) בקוד, צריך לטעון (ואולי גם לקמפל) את כולו, דבר שעלול לקחת זמן רב. אם מרכיב קטן במערכת כזו נכשל, כל המערכת עלולה להיכשל.
2. **Micro Services** - מדובר על ארכיטקטורת תוכנה לפיה יישומים מורכבים בנויים מיחידות קטנות המתקשרות ביניהן באמצעות APIs. מערכת כזו מאפשרת מודולריות נרחבת, מפרידה לשירותים שונים כל מרכיב בקוד, כאשר רמת הצימוד ביניהם נמוכה. ארכיטקטורה זו יוצרת חוסן בקוד, מכיוון ששירותים יכולים לעבוד באופן עצמאי. לא מדובר על טכנולוגיה במקרה זה, אלא על אסטרטגיית תכנון מוצר  
     
   יתרונות:
   1. ה-TTM של פונקציונליות מרכזיות בממשק קצר.
   2. המרכיבים השונים ניתנים לשימוש חוזר בבניית פונקציונליות חדשה.
   3. תכנית הבדיקות מתכווצת, צריך לבדוק רק את החלקים שנעשה בהם שינוי.
   4. יש אפשרות להרבה צוותים/מפתחים לעבוד באופן עצמאי, כמעט ללא תלויות.
   5. שיפורים ותיקונים יכולים להתרחש כל הזמן.
   6. מאפשר עבודה בשיטת Agile.

חסרונות:

1. כאשר מבצעים בדיקות רחבות, על כל המערכת, הדבר עלול להיות מורכב. צריך תחילה לוודא תקינות של כל שירות במערכת על מנת שנוכל לראות תוצאות מהימנות של בדיקה מקיפה.
2. בעת הרצת תוכנה הפועלת עם microservice, יש לדאוג לסנכרון ותיאום בין השירותים השונים, כדי למנוע התנגשויות, timeouts, בעיות read/write וכדומה.
3. ארכיטקטורה זו מחייבת את הארגון לשנות את המבנה הארגוני שלה, כך שיתאים לעבודה שוטפת עם שירותים כאלה. ככל הנראה, עובד אשר לא עובד עם ה-service לא יידע להשתמש בו.
4. קשה מאוד לבצע refactoring למערכת.

5) הפקודה היא:

hdfs dfs -put <localfile> <dest>

דוגמה:

hdfs dfs -put localfile /user/hadoop/hadoopfile